Universidade de Aveiro Departamento de Engenharia Mecânica
2017

Rafael Gomes Martins

Sistema de orientação automática para um vibrómetro laser.

Automatic orientation system for a laser vibrometer.

Universidade de Aveiro Departamento de Engenharia Mecânica
2017

Rafael Gomes Martins

Sistema de orientação automática para um vibrómetro laser.

Automatic orientation system for a laser vibrometer.

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Rui António da Silva Moreira, Professor Auxiliar do Universidade de Aveiro da Universidade de Aveiro e de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Apoio por parte do Centro de Tecnologia Mecânica e Automação (TEMA), através dos projetos UID/EMS/00481/2013-FCT e CENTRO-01-0145-FEDER-022083

O júri / The jury

Presidente / President

Prof. Doutor José Paulo Oliveira Santos
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar da Universidade de Aveiro

Prof. Doutor Rui António da Silva Moreira
Professor Auxiliar da Universidade de Aveiro

Agradecimentos / Acknowledgements

Agradeço à minha família por me ter proporcionado condições e apoio para que que este objetivo fosse concretizado.
Agradeço também aos professores Rui Moreira e Vítor Santos pela orientação e motivação dadas durante o semestre.
Por fim, agradeço também aos meus colegas e amigos que contribuíram para a minha formação pessoal e académica.

Palavras-chave Orientação automática; Vibrómetro; Calibração; Arduino; Galvanómetro.

Resumo

O estudo das propriedades dos materiais é muitas vezes realizado a partir da observação do seu comportamento sobre determinados estímulos, tal como vibrações. Estas vibrações podem ser medidas com recurso a um vibrómetro laser sendo que este dispositivo emite um feixe laser e efetua as medições com recurso ao efeito de doppler. Tal dispositivo está disponível na Universidade de Aveiro, sendo este um vibrómetro de ponto único, o que significa que a direção do feixe laser é a direção do vibrómetro, sendo que para obter uma medição noutro ponto é necessário mover o dispositivo. De modo a ultrapassar essa limitação, foi criado um dispositivo onde é incorporado o vibrómetro laser de modo a direcionar o feixe laser para posições pré-definidas. A escolha dos pontos para medição é feita com recurso, a uma câmera Kinect que atua como um scanner 3D. Para deflectir o feixe laser proveniente do vibrómetro foi usado um galvanómetro de espelhos cujo controlo é feito a partir de um Arduino Uno que comunica com o software em MATLAB através de comunicação serie. São analisadas diversas metodologias de definição automática de malha de pontos.

Keywords Automatic Orientation; Vibrometer; Calibration; Arduino; Galvanometer.

Abstract The study of material properties, is in many times done by observation of the material behavior under some vibrations. This vibrations can be measure with a laser vibrometer which has a laser beam that measures using the doppler effect. This device is available at University of Aveiro, but it is a single point vibrometer, which means that the beam direction is the same as the vibrometer, meaning that, to measure another point it is necessary to move the device. To avoid such issue, it was incorporates, on a system on which a device that deflects the laser beam into desirable locations was added. The process of choosing the points is made using a Kinect which acts as a 3D scanner. To deflect the laser beam of the vibrometer, a galvanometer was used which control is made by an Arduino. The Arduino communicates with MATLAB via Serial communication. Several numerical strategies to define automatically a measuring mesh are tested in order to identify their limitations and advantages.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter a brief introduction to the vibrometry field is discussed as well as the context of the project in comparison to existing products.

1.0.1 Vibrometry

Vibration is a mechanical phenomenon whereby oscillations occur about an equilibrium point. [70]

Vibration may influence the durability and reliability of machinery systems or structures and cause problems such as damage, abnormal stopping and disaster. Vibration measurement is an important countermeasure to prevent these problems. [71]

Usually to measure vibrations, accelerometers or laser vibrometers are used. The difference to consider is that accelerometers measure absolute structure movements and laser vibrometer measure differential displacements between its optical head and the object, so vibrations of vibrometer head support must be considered with care in choosing its location and in the subsequent data analysis. [72]

Figure 1.1: Kanomax 4200 accelerometer[88]

The biggest advantage of laser vibrometers in comparison with the accelerometers, is that it provides non-contact measurements.

The vibrometry technology is based on the Doppler-effect, sensing the frequency shift of back scattered light from a moving surface.[4]

The working principle of a Laser-Doppler vibrometer is optical interference. This means that two coherent light beams are required, one for reference and the other for

measurement.[4]

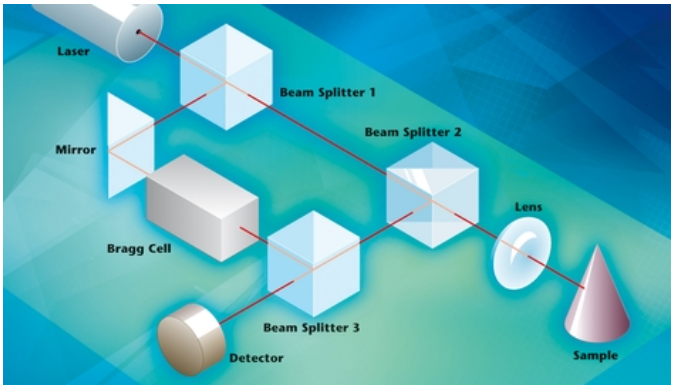


Figure 1.2: LDV principles[4]

The helium neon laser beam, is split by a beamsplitter (BS 1) into a reference beam and a measurement beam. Then, after passing through a second beamsplitter (BS 2), the measurement beam is focused onto the object under investigation, which reflects it. This reflected beam is now deflected downwards by BS 2, and then merged with the reference beam by the third beam splitter (BS 3) which is then directed onto the detector.[4]

1.1 Related work

In this section, several vibrometer systems are described. All vibrometry systems mentioned require a user to manually create the points of measurement since the system can not recognize the shape of the object and its location.

1.1.1 Vibrometer Products

Polytec Scanning Laser Vibrometer PSV-400

The PSV-400 system is a capable of measure different points without changing the position of the vibrometer of the object to measure. The system is composed by multiple components.

Figure 1.3: PSV-400[87]

The electronic unit is placed on a road case for easy transportation and consists of three components. On the top, there is the junction box for connecting external signals, triggering signal outputs. In the middle is the vibrometer controller which is the core of decoding the laser signals and in the button, is the data management system which is an industrial computer that runs the PSV software.[87]

Then there is a TFT monitor for displaying the analysis and set up of the mesh. The scanning laser vibrometer head has a build in video camera and a high precision scanning unit which allows for deflecting the laser beam to the object. The scanning vibrometer head is on a tripod for easy setup.[87]

To start a measurement, it is necessaty to position the scanning head in front of the object, in a distance from 80 mm up to 30m. [87]

The object will be visible on the TFT monitor in real time, and it is necessary to manually create the scanning grid on top of the objects surface.[87]

The mesh can be adjustable but the adjustments are manual, all using the Polytec's software. Then the measurement procedure can begin, the signal generator excites the object and in the TFT monitor, for each point, it is possible to see a color representation of the quality of the signal-to-noise ratio.[87]

For low frequencies, it is possible to scan 30 to 50 points per second.[87]

Politec RoboVid

Polytec states that RoboVid is a solution to remove many of the limitations of traditional contact transducer methods. [86]

Figure 1.4: RoboVid[86]

It consists on a 3-D scanning vibrometer mounted on a multi-axis industrial robot. [86]

RoboVid is engineered to be a stable, auto-configurable 3-D vibration measurement station for whole-body vibration mapping of complex-shaped objects. This combination of technologies can reduce test times for experimental modal analysis from weeks to days and from days to hours. The points to be measured can be derived from Finite Element models, thus facilitating model updating.[86]

The system uses as robotic manipulators the KUKA KR 120 and the R3500K Prime, and in each one is mounted a PSV-500-3D Scanning Vibrometers which has three scanning heads.[86]

The selection of the measurement points is done using the CAD model of the object that needs to be imported.

VibroMet MB-LDV

The VibroMet MB-LDV is, according to VibroMet, the only laser Doppler Vibrometer capable of multibeam scanning.[85]

Figure 1.5: MB-LDV[85]

This means that it can perform multiple readings at the same time, typically 16, with extremely high sensitivity and has a customizable beam pattern.[85]

Since all the multiple points are measured at the same time, transient events can be accurately measured. In addition, the relative phase between measurement locations can be used to generate modal vibration patterns in a single shot.[85]

It consists, in an optical head, an electronic controller and the computer and software.[85]

1.1.2 Laser vibrometer applications

Between many vibrometry applications, the ultrasonic fatigue testing of metals is one of the applications on which the product created would be used, making important the creation of a mesh of measurement points.

3D scanning vibrometry into the field of ultrasonic fatigue testing of metals

Since many engineered systems, such as heavily stressed motor parts or off shore structures must resist more than 10 million cycles due to either high frequency loading or a lifetime parts of up to more than 30 years. This cycle is named the Very High Cycle Fatigue (VHCF) regime. For a reliable application of those components, it is necessary to have de detailed knowledge of their material behavior in VHCF regime. Conventional testing facilities can only perform long duration tests at frequencies of up to 200 Hz.[1]

To test for example 10^{10} cycles in a short period of time, an ultrasonic testing facility for tension-compression experiments was developed at the Institute of Materials Science and Engineering (WKK) at the University of Kaiserslautern in Germany. The loading principle of the testing system is based on a piezo-electric converter, which is designed to

resonate fatigue specimens at a frequency of 20 kHz with a standing longitudinal wave that causes fatigue in the material. The eigenfrequencies are an essential property of the specimen is calculated. Finite element analysis is used during the design process in order to ensure an adequate specimen design. The 3-D scanning laser vibrometer promised to be an effective instrument to measure the natural frequencies and mode shapes and verify our finite element model.[1]

Since the standard techniques such as strain gauges have a measurement tactile nature, because of the high frequencies they become very difficult to use, and since the 3D scanning laser vibrometer performs non-contact measurements, it is the ideal candidate for the stress strain measurements during ultrasonic fatigue.[1]

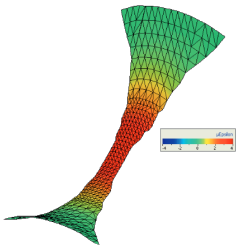


Figure 1.6: Strain distribution[1]

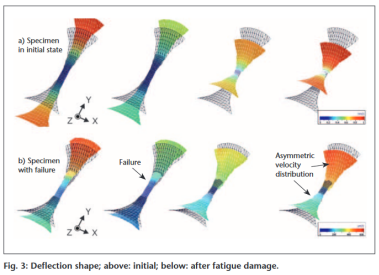


Figure 1.7: Deflection shape; above: initial; below: after fatigue damage[1]

1.2 Project Context

This project has the objective of developing a system for an available vibrometer, which is the OFV-505.

Figure 1.8: OFV-505 Vibrometer and Vibrometer Controller[74]

Table 1.1: OFV-505 Sensor Head specifications

Optical specifications	
Laser type	Helium Neon (HeNe
Laser class	Class 2, <1 mW, eye-safe
Laser wavelength	633 nm, visible red laser beam
Focus	Auto
Maximum stand-off distance	Up to 300 m (with OFV-SLR, surface dependen
Weight	3.4 kg
Typical spot size in mm at 1km	0.062

The interferometric sensor head works with a controller, the OFV-5000 which allows frequency, velocity and displacement measurements in real time.

It has analog and digital decoders that give a frequency range from near DC to 24MHz, with velocities up to approximately 10 meters per second and displacements from the picometer to meter range.[74]

This vibrometer is a single point vibrometer, which means that, in order to be used in ultrasonic fatigue testing, it is necessary to deflect its laser beam into the different measurement points.

In order to avoid damaging the vibrometer, a laser pointer was used to simulate the laser from the vibrometer.

Since the existing products always require an operator to create the mesh of measurement points, this system has as requirement, the ability to detect the objects shape and create the mesh on its surface, also,to avoid missing zones of an object of measurement, the mesh should take into account the three-dimensional shape of the object.

1.3 Objectives

Knowing the context of the project, multiple steps are necessary to produce the expected result, which is the creation of a system, compatible with the OFV-505 vibrometer, that can create a mesh of measurement points automatically. These steps can be summarized into the following topics:

- detection of the objects shape, 2D and 3D;
- creation of a mesh on the objects surface;
- development of a GUI to allow the user to choose the number of measurement points;
- deflect the laser beam;
- control of the laser beam deflection device;
- development the communication between the GUI and the beam deflection device;

Chapter 2

Experimental Setup

This chapter presents a detailed description of the components and software used on this work, allowing a better understanding of their limitations and their choices in order to fulfil the objectives described on Chapter 1.3, as for each objective, a component is presented.

2.1 Detection of the objects shape and creation of a mesh on the objects surface

To detect the objects shape, a camera is required, and since the three-dimensional shape of the object is to be taken into account, the Kinect Xbox360 as it was the only one available on the laboratory.

The Microsoft Kinect Xbox360 is developed by Microsoft for its gaming station, the Xbox. Since this camera can be found at a low cost compared to its industrial equivalents, it is largely used in scientific research[75].

Figure 2.1: Kinect for XBOX 360[76]

This camera has the following sensors:

- an infrared projector;
- an infrared camera;
- a RGB camera;
- a four microphones array;
- a tree-axial accelerometer.

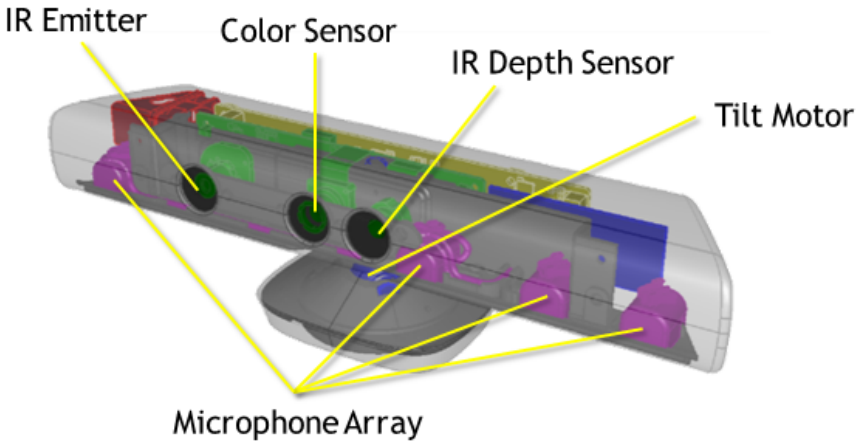


Figure 2.2: Kinect Sensors locations[9]

The infrared projector and infrared camera work together to construct the 3D point cloud that is essential for this work, and the RGB camera provides the colour image. The 3D image feed, taken with the depth cam, is produced inside the Kinect hardware and takes some seconds to start producing a valid result. The amount of time is not always the same but it usually takes around four to six seconds. The Kinect, even if connected to power does not start producing a point cloud, it only does it when one of the cameras, the RGB or the depth cam are requested, which means that only then, the three-dimensional image is produced, producing a four to six seconds delay.

The delay was found on the Kinect for the Xbox360, but, since the Kinect for Windows is the official sensor for windows compatibility, its performance can be different from the one mentioned. The camera, since it is the Xbox sensor, requires a special connector for 12V power and to connect to the usb port of the computer.

The Kinect sensor can only produce indoor measurements, since, the noise from the Infrared radiation of the sun prevents the Kinect infrared camera from seeing the projected pattern of the infrared projector. Since the work has the possibility of being introduced on the foundry industry, the Kinects potentialities of dealing with extreme temperatures were tested. For these tests, a part similar to those that the Kinect could find on daily bases on this industry, was inserted in an oven at two hundred and fifty Celsius degrees and after twelve minutes at these temperatures, it was removed and placed in front of the Kinect sensors at a distance of one meter.[91] The result was the same as if the object was at room temperature when the point clouds were compared. This means that the temperature of the object does not influence the Kinect depth measurement, at least within the mentioned range of temperatures.

To test the Kinect potential, Microsoft developed the developer toolkit which can run code that allows a visualisation of the infrared pattern, facial recognition, play games with voice commands, among other features.

With the Kinect accelerometer it is also possible to know which is the camera position according to gravity which then, working with a mechanical gear on the Kinect foot, allows angle adjustment according to the Kinect Y axis.

2.1.1 MATLAB

To create the program which then uses the Kinect as a 3D scanner, MATLAB is used, which is a software solution used by millions of engineers and scientists worldwide. Its platform is optimized for solving engineering and scientific problems since its matrix based, that is the natural way to express computation. It has built-in a vast library of toolboxes that helps the developement of complex solutions requiring data management and visualization.

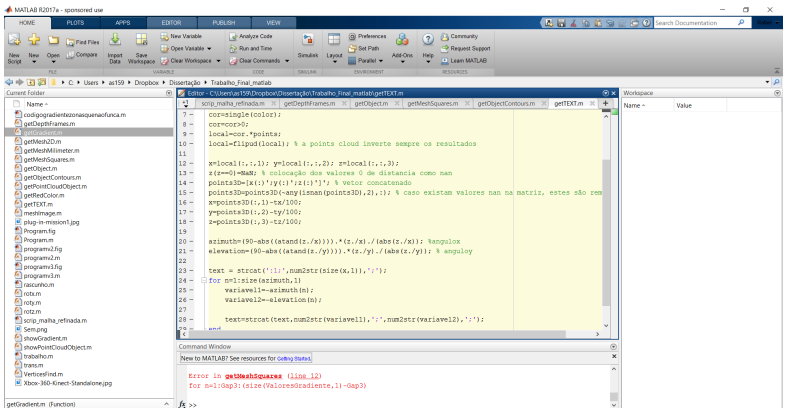


Figure 2.3: Matlab Environment

It was the platform used, were the code to observe the part and to create the measurement points was written.

The R2015a version was the first MATLAB version on which the work was started. For this project, since it is necessary the use of a Kinect, an add-on needed to be downloaded, which is the Kinect for windows add-on. This add-on allows then access to a vast group of specific Kinect functions, which are the camera's elevation angle, activation or deactivation of skeleton tracking, view data from accelerometer and videos, between others. This version however had some incompatibilities with newer functions, specific for point cloud treatment, as for example the pcfromkinect which allows as one of the input parameters the point-cloud color. The pcfromkinect equivalent, did not allowed the color image, as it only accepts the depth image and the depth device information. The mentioned function was only introduced on MATLAB R2015b.

To solve the incompatibility issues found on the 2015a version, the 2017a version was the version used, ensuring access to the point-cloud functions mentioned just above.

2.2 Development of a GUI to allow the user to choose the number of measurement points

GUIs (also known as graphical user interfaces or UIs) provide point-and-click control of software applications, eliminating the need to learn a language or type commands in order to run the application[80].

As the code was written using MATLAB, its GUI creation software was chosen to integrate that code.

2.2.1 GUIDE

GUIDE is the MATLAB tool to develop graphic user interfaces. It provides tools to design user interfaces for custom apps.

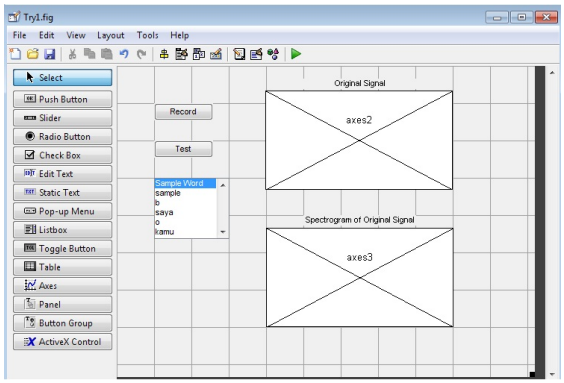


Figure 2.4: GUIDE Environment

GUIDE automatically creates a script with the call-back functions for each button added on it, allowing an easy integration of the visual and the functional component of the GUI. The call-back functions created, work as regular functions on MATLAB, which means that they do not save information on the workspace. So, to send information between the functions, its necessary to define global variables.

This program also creates the code to initialize the GUI, however, since MATLAB is a programing language that is not known for speed, the GUI takes some time to initialize. However, after starting, the GUI speed is the same as if the code was being executed outside the GUI.

2.3 Draw the measurement coordinates pattern with the laser beam deflection device

2.3.1 RGB-SCAN20 close-loop scanner

The RGB-SCAN20 close-loop scanner is the galvanometer kit used in this work with the purpose of deflecting the laser beam light into a desirable coordinate. It consists in the following equipment[23]:

- one galvanometer frame with two servos, each one with a mirror;
- two control drivers for the servos;
- one power supply +/-15V@100mA;
- some wires for the kit connections.

Each scanner component was drawn using Solidworks as it allowed easier measurement for construction of a prototype assembly.

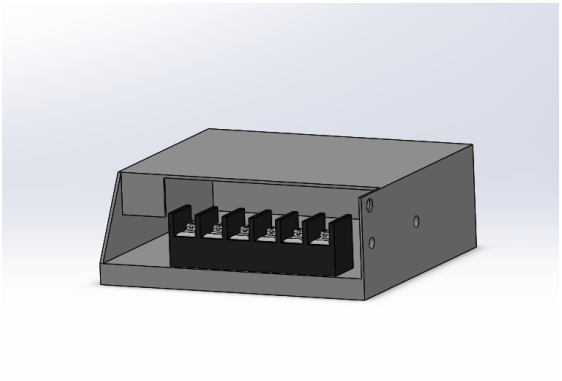


Figure 2.5: Power Source

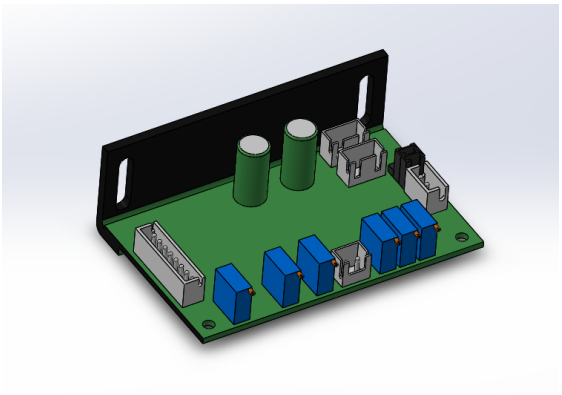


Figure 2.6: Drivers

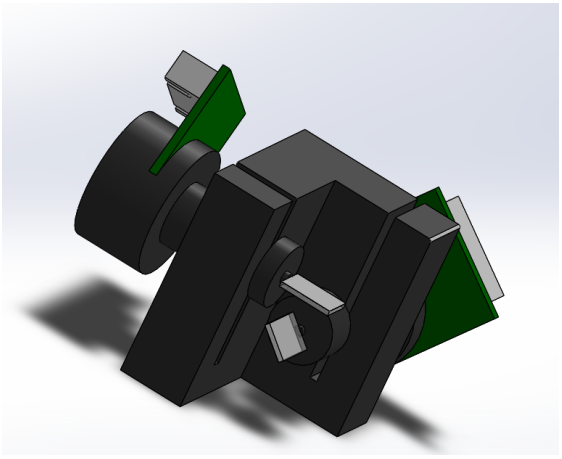


Figure 2.7: Galvanometer

The control of the servos of each axis is made using the control drives, as they transform the original current into movement of the mirrors. The scanner speed depends on the angle of deflection intended, as it can scan 20 thousand points per second (20

kpps) at 20 degrees of maximum deflection.

Table 2.1: Deflection Angles [23]

Delflection angle	Operating voltage	Speed@ Mirrors size
20 optical deflection	+/-15V	20Kpps @ 7*11*0.6
15 optical deflection	+/-15V	22Kpps @ 7*11*0.6
10 optical deflection	+/-15V	25Kpps @ 7*11*0.6
8 optical deflection	+/-15V	30Kpps @ 7*11*0.6
5 optical deflection	+/-15V	35Kpps @ 7*11*0.6

By observing the previous table, the maximum number of points per second is 35000 but, to achieve this result, the mirrors could only rotate a maximum of 5 degrees. Since the project pretends to have a wider range, the 20000 points per second is the chosen result with a maximum angle of 20 degrees. The galvanometer Kit is controlled using voltage from -5 to 5 VDC requires current from minus fifteen to fifteen volts to work which is supplied by the power source.[23]

The control signals for the drivers are displayed on the next table:

Table 2.2: Control Signals[23]

Power input			
XH-3 Connector pins	Description	Remark	Cable color
3	+VCC	+15V/1.0A	Red
2	GND		Black
1	-VEE	-15V/0.6A	White
Signal Input			
3	Control signal +	-5V∼+5V analog sig.	Red
2	S-GND	Ground	Black
1	Control signal -	-5V∼+5V analog sig.	White

The ground of the signal input acts as a mass, being the end result a mean of the first and third pins. This galvanometer is to be operated between 0 and 50 degrees and has a reacting time of 100ms.

2.3.2 Galvanometer applications

There are several applications using galvanometers for laser scanning purposes. Some examples are presenter herein, where the use of a pair of galvanometers provide the effect desired for our application.

Galvanometer scanning technology for laser additive manufacturing

A galvanometer laser beam scanning system is an essential element in many laser additive manufacturing technologies including Stereolithography (SLA), Selective Laser Sintering (SLS) and Selective Laser Melting (SLM). To achieve a high quality finish part it is necessary to deliver a uniform laser density on the powder bed.[6]

Xi Luo[6] also says that in order to achieve the intended finish quality, it is necessary to modulate the laser power as function of the scanner velocity during the acceleration or deceleration periods. Another strategy is to use smart control algorithm to maintain the constant speed throughout the job.[6]

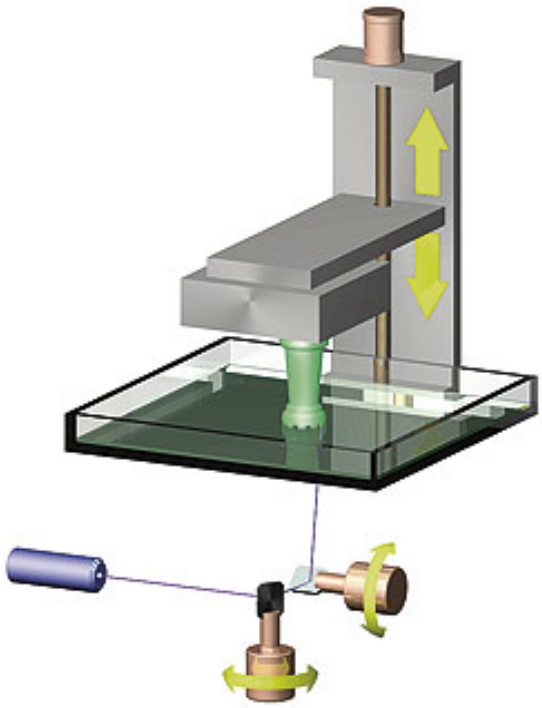


Figure 2.8: SLA printer principles[7]

Xi Lou also mentions that as part of the ongoing effort of the biomedical imaging community to move Optical Coherence Tomography (OCT) systems from the lab to the clinical environment and produce OCT systems appropriate for multiple types of investigations in a medical department, handheld probes equipped with different types of scanners need to be developed. These allow different areas of a patient body to be investigated using OCT with the same system and even without changing the patient position.[6]

Optical-resolution photoacoustic microscopy for imaging blood vessels in vivo

According to Yi Yuan[8], an optical-resolution photoacoustic microscopy system was designed and fabricated by integration of a two-dimensional scanning galvanometer, with objective lens, unfocused ultrasound transducer and a sample stage for imaging blood vessels in vivo. The lateral resolution of the system was measured to be arround 5

micrometers.

The system has the objective of monitoring blood vessels of mouse ears. The experimental results showed that galvanometer-based photoacoustic microscopy holds clinical potential in detecting lesions of blood vessels.[8]

2.3.3 Control of the galvanometer

In the designed solution, a low cost and flexible control system is desired, providing the required flexibility to be modified and adjusted along the design and assessment process. For that purpose, and Arduino UNO development platform was considered.

Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328p. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz internal clock, a USB connection, a power jack, an ICSP header and a reset button.[13]

The Arduino Uno is the device responsible for the control of the galvanometer servos. This Arduino can communicate using at least serial communication and I2C communication, as the I2C communication is achieved using the A5 and A4 pins, being the fifth analog pin the timer pin and the fourth the data pin. It is also possible to connect a maximum of 12 volts to the Arduino using the power jack. The output of those 12 volts is on Vin pin of the POWER section, allowing the Arduino to be used as power source of equipment that uses more voltage. The serial communication baud rate can variate from 300 to 115200 bits per second, so, since the program needs to send large strings, the maximum speed was used.[14]

The total memory of the Arduino Uno is 32 KB of which 0.5KB are used for the bootloader, which creates a small barrier for the construction of large programs.

Figure 2.9: Arduino Uno[90]

2.3.4 Arduino IDE

To program the Arduino Uno, it was necessary to use Arduino IDE to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software[79].

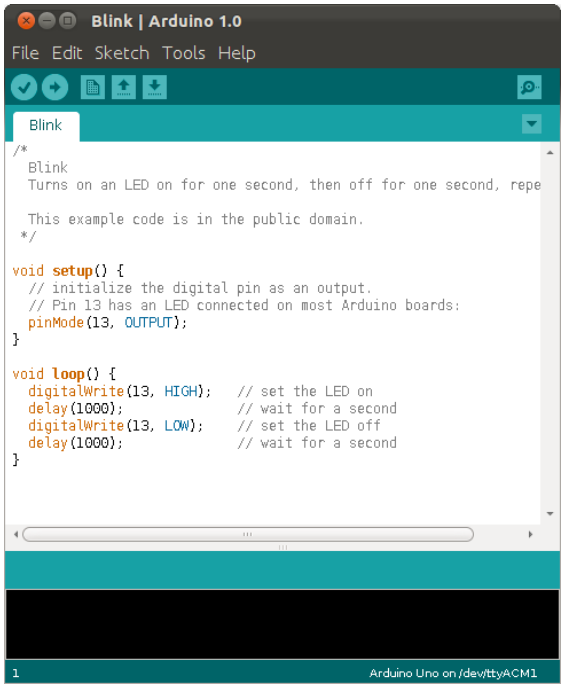


Figure 2.10: Arduino IDE Environment

The Arduino uses, three main functions, setup[53], loop[54] and serialEvent[55]. In the setup, the variables are initialized with a value and the communications are started. The loop function, as the name indicates keeps the code in a loop. The serialEvent function is actived when the Arduino receives a message.

2.3.5 Analog signal creation

The Arduino can only output a digital signal, and since the galvanometer needs an analog signal to be controlled, a way to generate the desired signal was developed. For this effect DAC was the ideal device to use but since it was not available at the moment, digital potentiometers were tried. Then, as the DACs took some time to arrive, PWM was also tried.

Digital Potentiometer X9C103S

The digital potentiometer X9C103S was the first solution considered to ensure an analog signal connection to the galvanometer drivers.

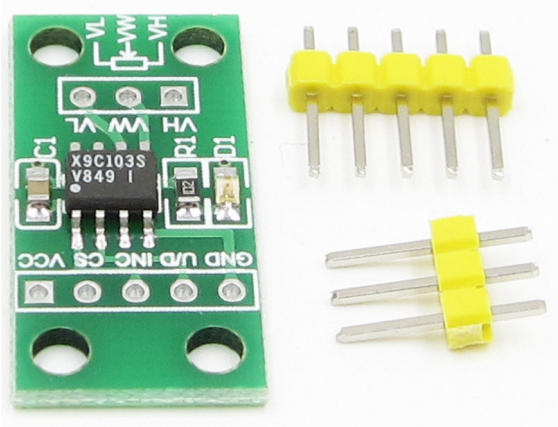


Figure 2.11: Digital Potentiometer X9C103S

The potentiometer resistance is 10kΩ and it has 99 resistive elements which means that it can create 100 variations of the signal between the 0 volts (minimum) and the 5 volts (maximum)[77].

In Figure 2.11 the potentiometer pins can be seen. The pins VL, VW and VH, work just like in an analog potentiometer. The VL is connected to ground and the VH to 5V. The VW is the output analog signal which is controlled by the other pins U/D, INC and CS. The VCC and GND are connected to 5V and ground respectively.

The way U/D (up, down), INC (increment) and CS (chip select) are controlled is as it follows:

to add one step, CS needs to be low, U/D need to be high and then INC is set low to increment one step;

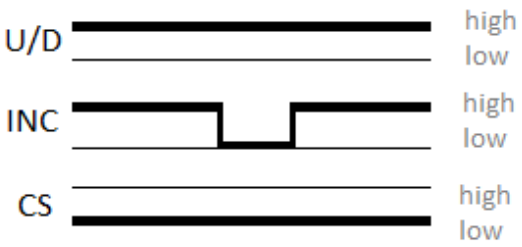


Figure 2.12: Add one step to the digital potentiometer

to subtract one step, CS is still low but now U/D needs to be low, then low on INC is sent;

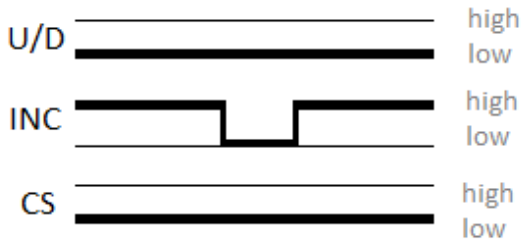


Figure 2.13: Subtract one step to the digital potentiometer

to store the wiper value for the next time the device is turned on, U/D and INC both need to be high and then you set CS high;

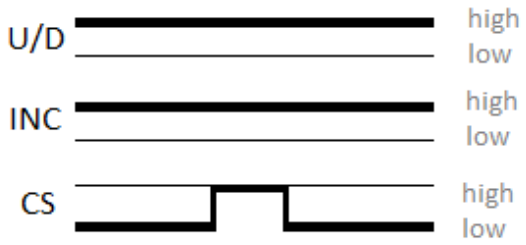


Figure 2.14: Save value for the next time the device is turned on

if the CS value is high in any of the previous control steps, nothing happens because the device enters standby low power mode[78].

The variations take time to perform and since the potentiometer works in steps, first a library with the relation voltage-steep values would need to be created and then the intended voltages would need to be compared with the library to know how many steps were necessary. The values for each step, can be found at appendixA. Each operation on the potentiometer takes time, so it is hard to calculate how many points per second the potentiometer was capable of, but doing an estimative, as each change on the INC takes 1 millisecond[77], and it needs to be there one microsecond to be activated it would result in 60 000 changes per second. The change in the up/down value would also be necessary to perform, but since it only takes 100 nanoseconds, the value 60 000 is a very close estimative of the final number of points.

Due to the speed restriction, the potentiometers were not used.

Motorola MC74HC00AN

Other way to control the galvanometer was with of PWM signals, which, using a Motorola logic gate reduced the number of used PWM pins to two. This two pins control two gates each, allowing a quick transition between each value.

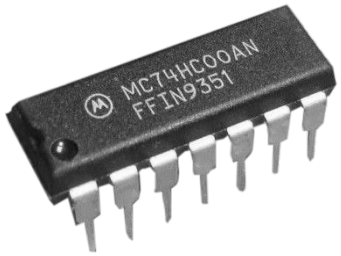


Figure 2.15: Motorola MC74HC00AN [18]

Table 2.3: Motorola MC74HC00AN specifications [18]

	Value	Unit
Logic Function:	NAND	-
Logic Family:	74HC	-
Number of Gates:	4	Gate
Number of Input Lines:	2	Input Line
Number of Output Lines:	1	Output Line
High Level Output Current:	- 5.2	mA
Low Level Output Current:	5.2	mA
Propagation Delay Time:	75	ns
Supply Voltage - Max:	6	V
Supply Voltage - Min:	2	V
Maximum Operating Temperature:	+ 125	degrees C

SparkFun I2C DAC Breakout - MCP4725

The last control method uses digital to analogue converters "DAC". The DAC, of the Data Logging shield, is the MCP4725 model of SparkFun. Four DACs had to be used to convert the four digital signals required, one for each axis signal, positive and negative.

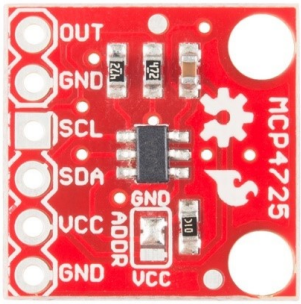


Figure 2.16: MCP4725[16]

Table 2.4: MSP4725 Specifications[16]

	Value	Unit
Fast Settling Time	6	microseconds
Operating Voltage Range	2.7 to 5.5	V
Power Up Time	2.5	microseconds
Operating Temperature Range	-40 to 125	degrees C
External A0 Address Pin	can be tied to VDD or VSS	
Resolution	12	Bits

To the SLC and SDA a connection to the Vcc pin was done using a 1kΩ resistor in order to keep the lines high because of active low devices in the I2C line. [17]

The I2C bus wires allow the connection between multiple devices using just two wires.

2.3.6 Arduino Shields

To use the previous components on the Arduino, shields were developed, which are boards that can be plugged on top of the Arduino. To these boards, components can be welded, making a PCB (printed circuit board) equivalent. Two shields were used on this work, one KEYES XD-204 Data Logging Shield Module and one Prototype Shield.

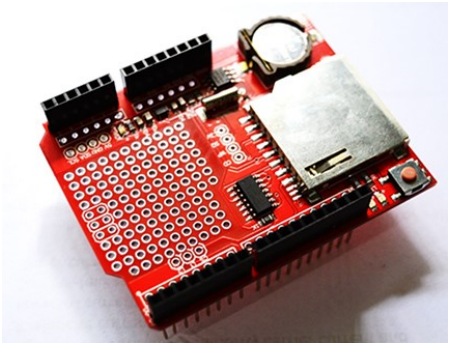


Figure 2.17: XD-204 Data Logging Shield Module

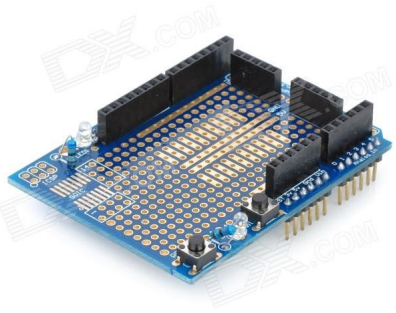


Figure 2.18: Prototype Shield

These two shields were made to operate two different methods of controlling the galvanometer, the "Data Logging" shield uses I2C with the DACs and the "Protoshield" PWM with the logic gate.

The programming and schemes for the shields is described on chapters 4.5.1 and 4.5.2 for the Protoshield and Data logging shield respectively.

2.4 Draw the measurement coordinates pattern

2.4.1 Lasers

The lasers have the job of replacing the vibrometer laser beam, and, draw the mesh coordinates.

The number of lasers on this work was three, all with different specifications.

The first laser used, was removed from a laser pointer and since its shell did not had any reference, it was impossible to find specifications for this laser.

Figure 2.19: laser shell

Figure 2.20: laser with k2 button bypassed

To use the laser, the button k2, which was the one that turned on the laser had its connections welded, as it can be seen on the previous figure, and the positive and negative connections also had one jumpers welded, to allow an easier connection to the Arduino.

Without references, tests were made to find the minimum on-off frequency for this laser, which consisted in the creation of an Arduino program that sent a high signal, followed by a pause, and a low signal, again followed by a pause. The minimum pause value was around 40 milliseconds, since the laser, even without passing through the galvanometer mirrors, which if dirty absorb some light, was barely visible.

This laser pointer however had a problem, that was only noticed when it was replaced, that was the fact of turning on and off in a timed interval, which was not measured. This on-off automatic switching, created several problems that first were attributed to the code, since this behaviour was not expected from the laser, and even drawing simple lines appeared as dashed lines. The problem was more visible when the mesh points were drawn, since they could not appear at the same time, instead, it looked like the galvanometer did not had enough speed to draw it.

The laser was then changed momentarily to another laser pointer, to allow further code development while the other laser pointers were delivered. This laser pointer has the expected behaviour of being always on which allowed to prove that the developed code was in fact correct,identifying the first laser as the cause of the problems observed.

Figure 2.21: Second laser pointer[24]

Since this laser pointer was used with duct tape on the on-off switch to be on, and was only temporary to this work, its specifications will not be discussed.

KY-008

The KY-008 laser module is made specially to be compatible with Arduino. It has three pins, the ground, the reference or input voltage and the on-off signal pins. The on-off signal is five volts to turn the laser on and zero to turn off the laser. [26]

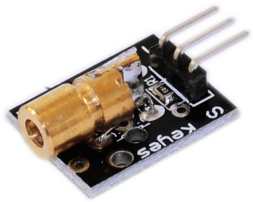


Figure 2.22: KY-008 alser module[26]

Table 2.5: KY-008 specifications[27]

	Value	Unit
Operating Voltage	5	V
Output Power	5	mW
Wavelength	650	nm
Operating Current	Less than 40	mA
Working Temperature	-10 to 40	degrees C

Since there is no official datasheet for this module[28], the centre pin outputs the same voltage as the switch pin, so it can be used as reference. On the other hand, information can be found that states that the central pin is used to feed 5 volts to the module, even though it works without the central pin being connected.

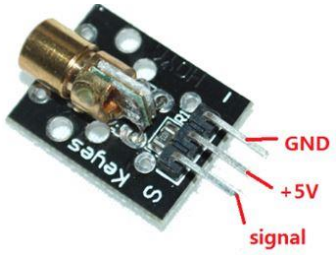


Figure 2.23: KY-008 connections[27]

Since at the time of construction of the shields, the only information found said that

it should have 5 volts on the central pin, the connection to the laser on the shields has 5 volts on the central pin. The following images represent a drawing using the project galvanometer with the KY-008 laser.

Figure 2.24: Pikachu[104]

Figure 2.25: Pikachu drawn using the galvanometer

G80 532nm 80mW

The final laser tested was an 80 milliwatt laser.

The laser is compatible with the Arduino, since the minimum power to work is five volts. It also has an internal fan cool the laser, and a parallel circuit board.

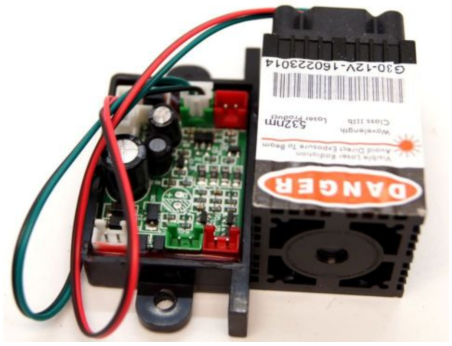


Figure 2.26: G80 532nm 80mW[30]

Since the reference of the laser could not be found online, its specifications could only be found on the retailer webpage.

Table 2.6: G80 532nm 80mW[30]

	Value	Unit
Input voltage range	5 to 12	V
Output power	80	mW
Operating temperature	10 to 35	degrees C
Wavelength	532	nm
Expected lifetime	5000	hours

This laser however is class 3B, which is the second most dangerous of laser classes. This laser class is used inside CD and DVD writers although the writer unit itself is class 1 because the laser light cannot leave the unit.

A Class 3B laser is hazardous if the eye is exposed directly, but diffuse reflections such as those from paper or other matter surfaces are not harmful, so to use this laser on the project, protective eyewear is advised.

2.4.2 Temperature Control

When using the galvanometer, it was noticed that the drivers generated heat, so, to study how much heat was being generated, a temperature sensor was used.

Temperature sensor LM35

Since the galvanometer drivers were placed on top shelf ceiling of the prototype, the heat generated gets converged in that area, and, to measure how much is the temperature on that area, a LM35 sensor was placed just under one of the drivers.

Figure 2.27: LM35[19]

After five minutes working, the temperature increased 10 Celsius degrees in the drivers area, and to extract the heat generated by the drivers, a fan was installed.

Cooler Master DF1202512RFUN

Measurements after the installation of the fan, showed that it was capable of returning the studied area to ambient temperature. The specification of the fan can be found at Appendix E.2.

Table 2.7: LM35 specifications[19]

	Value	Unit
Accuracy	0.5	degrees C
Temperature Range	-55 to 150	degrees C
Operating voltage	4 to 30	V
Self heating	0.08	degrees C
Scale Factor	10	mV /degrees C

Figure 2.28: Cooler Master FAN[20]

Pro K FAI12V-3A(C)

To power up the fan, which requires 12V to work, a power source was added. This power source also supplies the power to the prototype interior lights which are only for esthetics and its specifications are on appendix E.3. More information on the power source is available on appendix E.1.

Figure 2.29: Pro K FAI12V-3A(C)[22]

2.5 Mechanical Prototype

The first prototype, 3D designed, was no more than the creation of the components bases that would use the same tracks which were aluminum profiles profiles. The parts were produced in aluminum, and machined but since the assemly profiles did not arrived in the expected time another solution was developed.

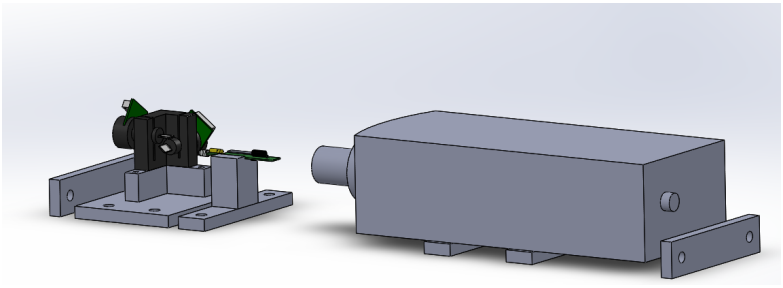


Figure 2.30: Temporary structure

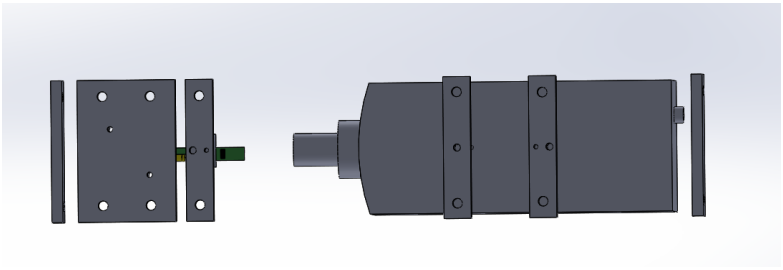


Figure 2.31: Temporary structure, bottom view

The other developed prototype was thought in a way where it must be able to incorporate all the components and allow an easy fixture of them. This way a first 3D drawing was done to allow a quick perception of the global dimensions.

Figure 2.32: Prototype Box - front view

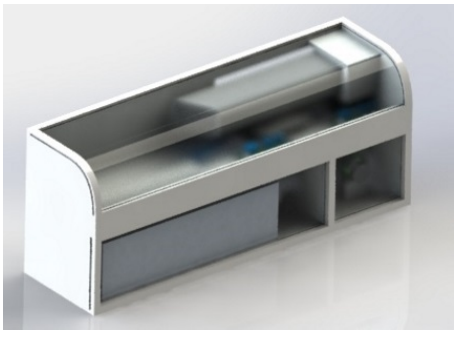


Figure 2.33: Prototype Box - back view

Since this design could only be produced using a CNC or any similar machine, a look alike prototype was then developed, using bigger dimensions than the ones on the previously described prototype.

This prototype, started as scrap metal, from an old exhibitor.

Figure 2.34: First Step of the prototype

The final look over the prototype can be seen in the following figures.

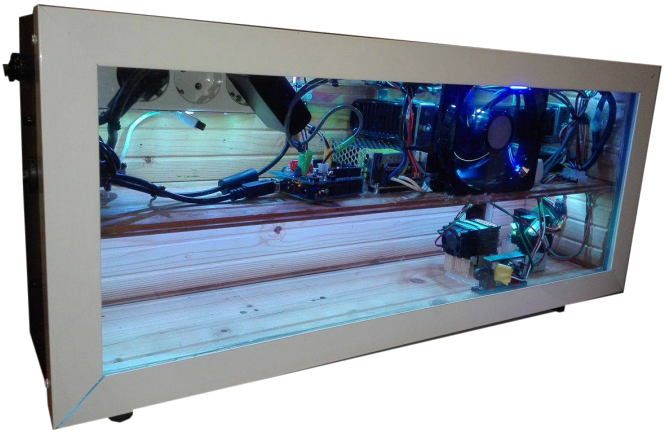


Figure 2.35: Back view

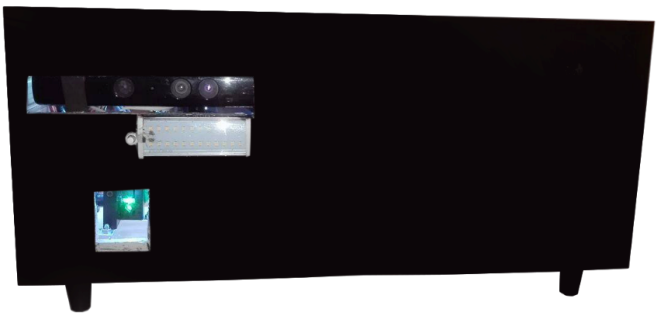


Figure 2.36: Front view

Information on the prototype construction and details can be found at appendix B.

Chapter 3

Creation and selection of the measurement point

This chapter discusses the creation of the points on which the vibrometer will perform the necessary measurements starting from the image acquisition with the Kinect.

3.1 Image acquisition

The image acquisition is made using one Kinect, so it is necessary to install an add-on to be able to use the Kinect, called, "Image Acquisition Toolbox Support Package for Kinect For Windows Sensor"[33] for it to work on MATLAB.

With the add-on installed, the access to the Kinect cameras is made using the command videoinput[34]. With this command, it is possible to choose between the infrared, RGB and the depth cam. The depth cam consists in the 3D image that results in the interpretation of the infrared pattern distortion.

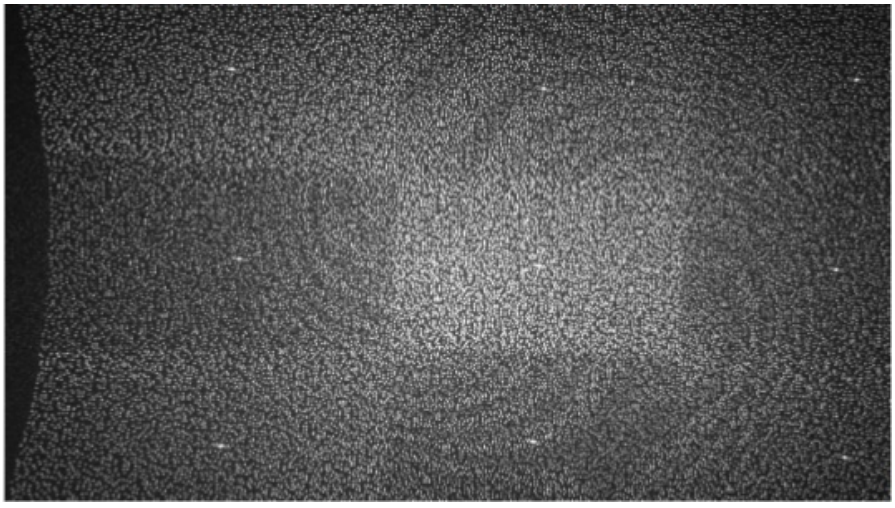


Figure 3.1: IR pattern

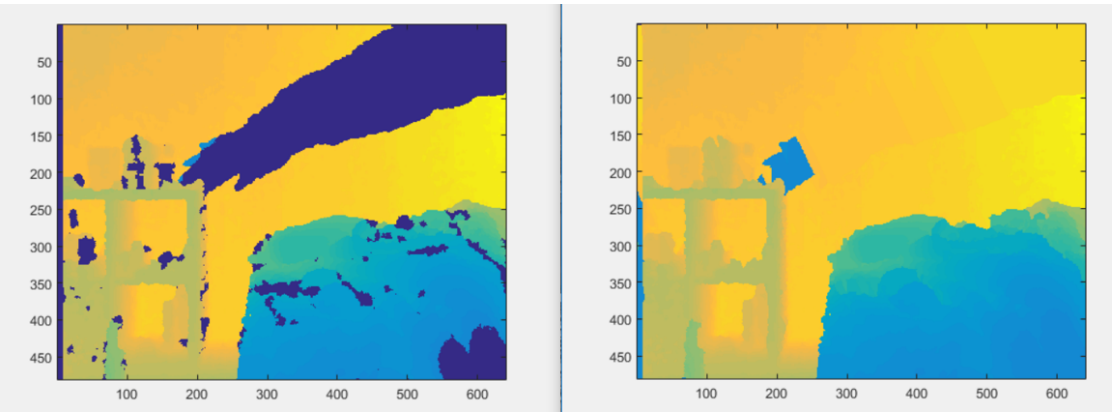


Figure 3.2: Kinect Depth Normalization

After getting the image feed, one RGB frame is saved using getsnapshot[35], and 21 depthcam frames are saved using the function getDepthFrames, which also uses the frames taken after the first one to complete any missing depth data. This missing depth data is represented by a dark blue in the Kinect depth cam preview, represented on Figure 3.2, and consists in zeros. The frames are saved along 10 millisecond intervals.

The need to implement the 21-frame method emerged with the observation of big chunks of dark blue (missing data) on objects with highly reflective surfaces. Another solution was first implemented, but, it modified the data from the original image in a way that it would be impossible to ensure that those values could be trustworthy. This solution is a function, found online, with the name Kinect Depth Normalization[36], and what it does is replacing the missing values with a mean of the surrounding values. As it can be seen in Figure 3.2, the big dark blue object was in fact the authors arm and hand. Since the arm was less than 80 cm from the Kinect, it appears as dark blue, as it was supposed to be. What the function did was assign values to that missing information. With the 21-frame method, this problem does not appear and its possible to get most of the missing information.

3.2 Selection of the Biggest Object

The selection of the object for the measurements is done using a threshold[37]. This simple solution can easily capture white parts in a black background. Before the threshold solution being implemented, since there was no parts to test, the biggest red object in the image was selected. The red-object solution was tried, supposing that, by separating the three RGB channels, red, blue, and green, it would be easy to capture the object. This solution had some difficulties identifying the objects since the luminosity conditions had a huge influence on the Kinect perception of the red color since, in low luminosity, conditions it has the tendency of seeing a reddish color. Even though these problems occurred, this solution was used in the early stages of the work. A red cardboard was used to simulate an actual object as it could be deformed into different shapes and curvatures.

The threshold solution that is now in place, can distinguish most of the colors in a black background with ideal luminosity conditions, making this the final argument for

its application on the program.

To use any of the solutions previously described, the RGB snapshot, is converted into a black and white image, with rgb2gray[38] command and the image is then binarized with the imbinarize command. The binarization operation is performed using the inverse of the black and white photo, with the purpose of initiating the background removing process.

To remove the background, the result of the binarization is duplicated to save one and apply transformations to the other. The first step to remove the background is to select the biggest area of background, using the regionprops command.

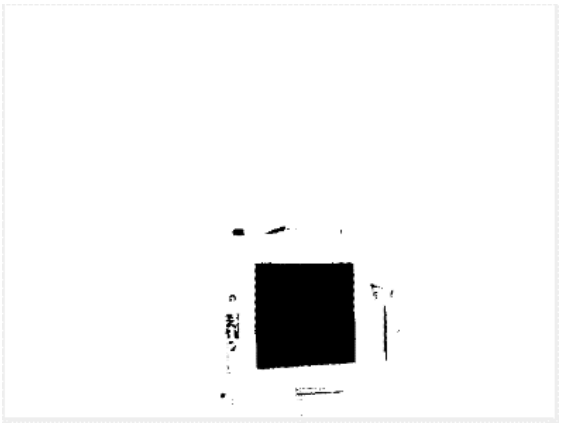


Figure 3.3: First stage of the background removal

Then the background is removed by doing the inverse of the previous image and performing a border cleaning operation with imclearborder command. This removes everything that was identified as object at start that is in contact with the border. This is useful, for example, if there are any white stripes in the ground as it happens on the department laboratory, LAR, or to remove any white walls on the background. The result of this operation is an image as represented in Figure 3.5.

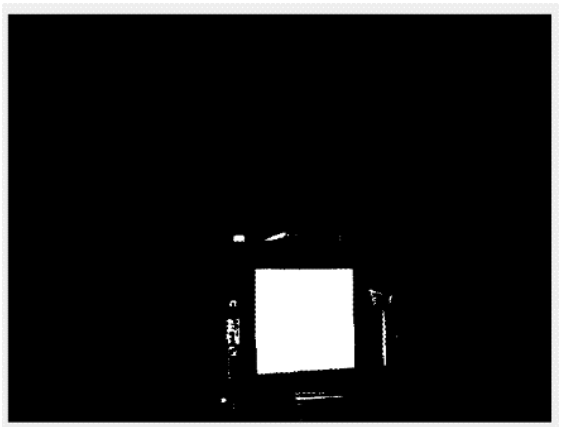


Figure 3.4: Second stage of the background removal

The next step is the removal of the noise left. To achieve this, again the biggest object

was selected with regionprops. The result, is in this case, is only the white big square. This result is multiplied against the negation of the image saved on the binarization mentioned before. This multiplication in this test-case was not necessary, since the object is a square without holes inside, as if there was any, this multiplication would add that information to the big square. The information was lost in the first place with the focus on the background, at the beginning of the process where the object in focus was the biggest background.

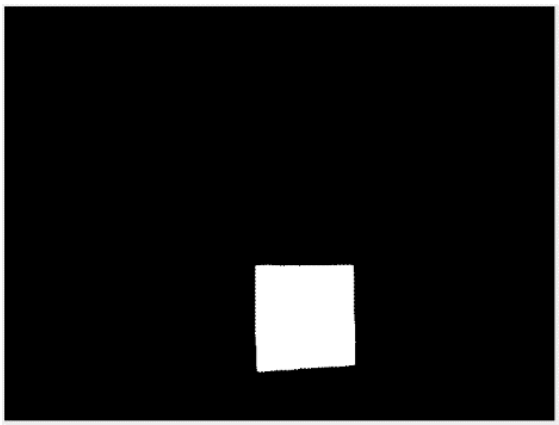


Figure 3.5: Final stage of the background removal

The final step is, again, the selection of the biggest object with the purpose of removing any possible noise that could be in the middle of the holes. To perform the task of extracting the objects location "getObject" function was created, with input parameters the color image and the desired threshold and the return of the function is the objects location just like the previous image, which is a logical image.

To use this image to obtain the objects point cloud, it needs to be a three-dimensional matrix, so the repmat command is used.

3.3 Point Cloud

The 3D image with the object's location is now used to replace the original RGB snapshot, to obtain a point-cloud with the object represented as white and everything else as black. This allows the extraction of the objects point-cloud information, whose calculation is done using the pcfromkinect command. This command accepts as parameters the depth device information, a depth snapshot and a RGB snapshot, being this RGB snapshot the black and white image of the previous paragraph.

Before the use of the black and white image as RGB snapshot, another way was tested. It was though that the depth image, multiplied by a logical black and white image with the position of the object, could be inserted into the pcfromkinect[39] function.

It was later discovered that the center of the depth image is not calibrated with the RGB image, being these calibrations done by the pcfromkinect command.

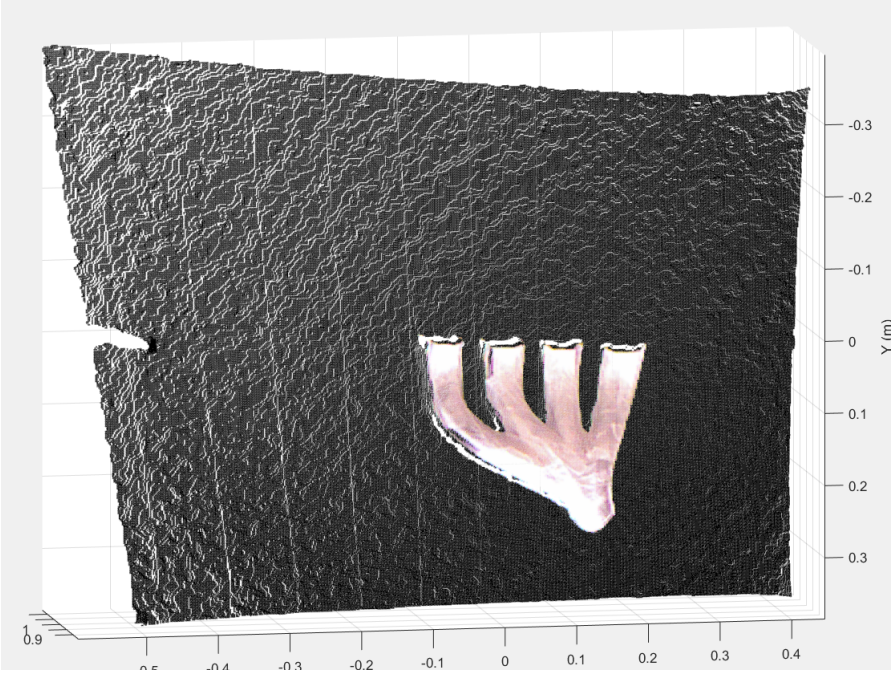


Figure 3.6: Point Cloud with object in white and background in black

The pcfromkinect calculations result in multiple point cloud parameters, but the only two relevant for this work are color and location. To get the real-world location of the object, first, the points bigger than zero where find, then those locations were multiplied by the point-cloud locations. The result is the 3D information of the object alone.

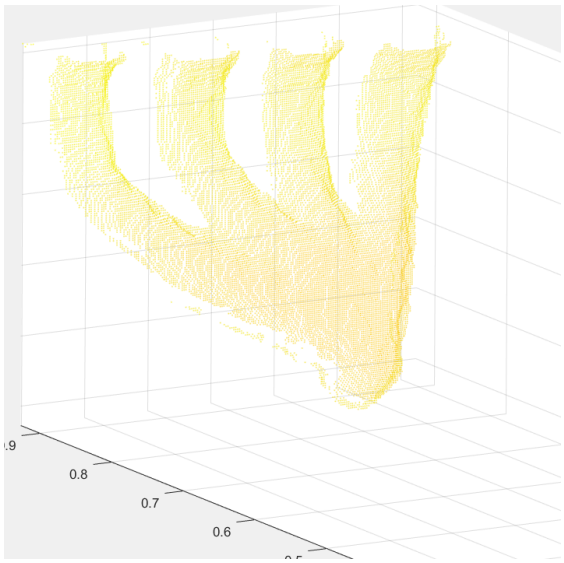


Figure 3.7: Point Cloud of the object

This point-cloud information is essential for to the three-dimensional mesh creation functions.

3.4 Two-Dimensional mesh creation

The measurement points are obtained with a mesh where the interior points are initially calculated and then the objects contour is transformed into points to complete this mesh.

The two-dimensional mesh is the simpler of the three methods for the creation of the interior points, since only the objects location information is used. This method starts with a black image. Then it adds equidistant points, in pixels, creating equidistant rows and columns. The result of this operation is a black image with white points, so to apply this mesh to the objects location, this image is multiplied by the objects location image. The result is an equidistant mesh in the objects location.

For the creation of this task, a function was built, with the name getMesh2D to compact the code. This function accepts the object location and the distance between points of the mesh, in pixels. This functions returns the image with the equidistant mesh in the object surface.

Figure 3.8 represents a 2D mesh that is zoomed-in to allow a visualization of the mesh.

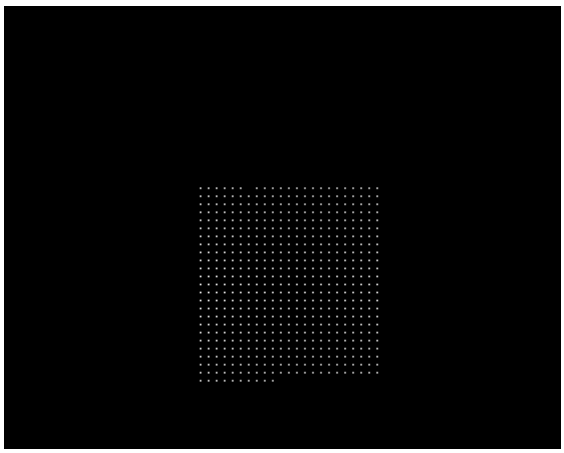


Figure 3.8: Two Dimensional interior mesh

3.5 Three-Dimensional mesh creation

The three-dimensional mesh creation is the interior mesh creation method that uses the point cloud information. The purpose of this method is to increase the number of points in a high gradient zone.

To find a way to create measurement points using the point cloud, two solutions were considered. These two solutions are very distinct in the way they solve the problem, one consist in the division of the image in multiple squares and calculate the medium gradient in each one, then five previous calculated meshes with different thicknesses, similar to those of the two-dimensional method, are used according to the medium gradient value. The other solution is based on the gradient distance, horizontally and vertically of each point.

3.5.1 Division based on gradient distance

The point cloud information, gives us a 3D matrix of the coordinates x, y, and z, of each pixel.

Then the gradient operation is performed, resulting in the normal vector matrix of the horizontal and vertical distances between points. This is not the real-world distances, instead is the mean between the previous and the next point. As the object is expected to keep a certain regularity in the changing of values, a bigger change will result in higher gradient. The gradient functions described are applied to all three point cloud layers, resulting in the x, y and z distances in the horizontal and vertical. With these six distances, two for each axis, vertical and horizontal gradients, the final horizontal and vertical distance between each point are then calculated.

This calculation is done using the square root of the sum of the squares of each matrix. The result is two matrixes, one with the horizontal distances and another with the vertical distances.

Using these distances, two cycles were created to read each row, one cycle for each matrix, summing each number until a certain value is reached, setting that coordinate as one in a zeros matrix. The result is represented in Figures E.4 to 3.11.

Figure 3.9: Horizontal selection

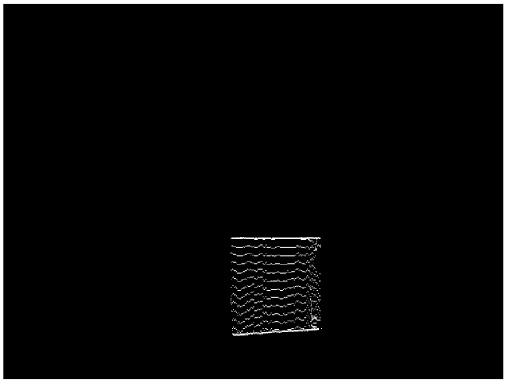


Figure 3.10: Vertical selection

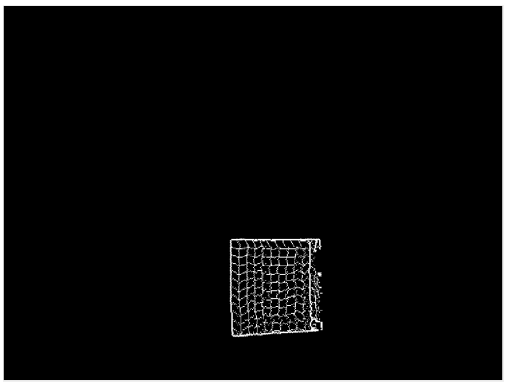


Figure 3.11: Sum of the Divisions

The previous figure shows the result of the sum of both matrixes. Visually, it is easy to see where the intersections of the mesh are, but the creation of code to perform this task is hard. The simpler and the one with better results was the assumption that the points where the matrixes "lines" intersect, had the value two, which is the sum of the two pixels.

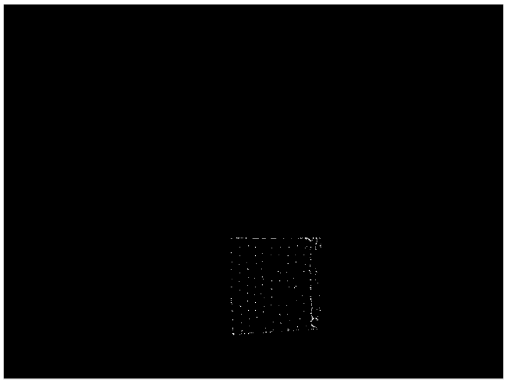


Figure 3.12: Bigger than one

This solution is very fast to compute but the results are incomplete as sometimes the pixels, where it was supposed to be an intersection, have a one or more pixels' distance. Other solution consisted on connecting white pixels to establish lines both horizontally and vertically. Then the points of intersection were the intersection of the lines with the command polyxpoly[40].

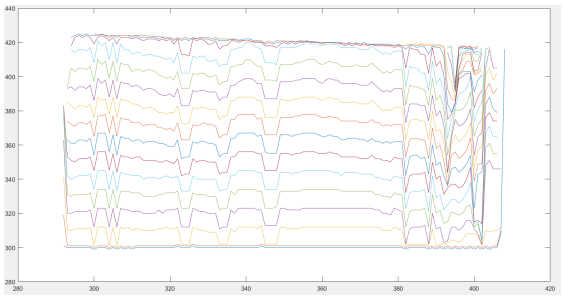


Figure 3.13: Horizontal lines

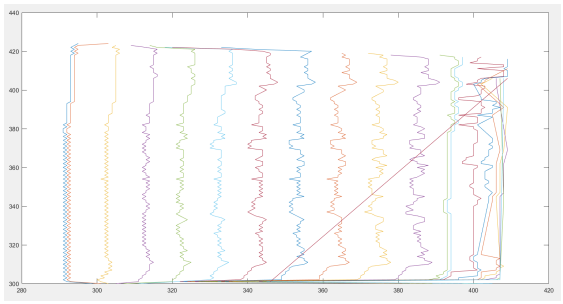


Figure 3.14: Vertical lines

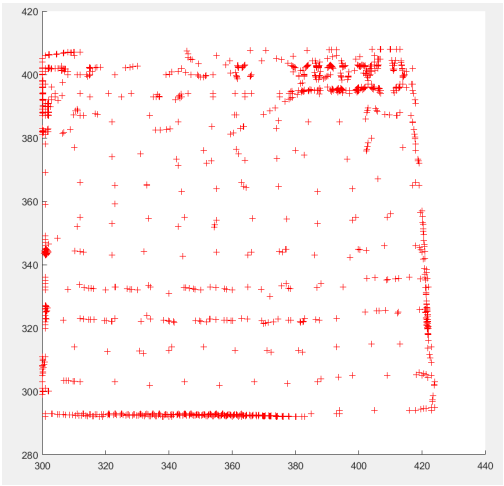


Figure 3.15: Sum of both lines

This operation, besides taking a couple of seconds to calculate, showed more intersection points than it was supposed to show, making this solution time costly for the result, making the first solution the best choice.

To perform this task, a function was created to incorporate the code that accepts as input parameters, the point-cloud, the object location, the new distance, and the adjustable distance. The adjustable distance is the previous mentioned distance that is set for the two cycles to create the internal mesh, the new distance is the selection of only

a few points from the previous mesh. The introduction of the adjustable distance has decided, after the verification that, if the adjustable distance was small, as for example two, by selecting points of the previous matrix in a certain interval it had a better accuracy than if it was set originally on the adjustable distance. This accuracy difference is caused by the probability of the pixels not intersecting for bigger distances. The output of the function is a black image with the intersection points as white.

The result of a function in a cardboard is represented in the following image.

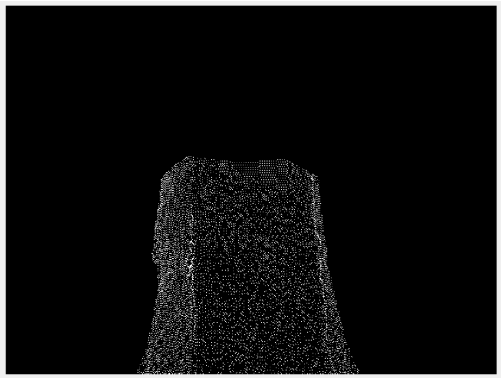


Figure 3.16: Interior mesh of Red cardboard with 5mm distance

3.5.2 Division in squares

The division in squares is the alternative 3D method. This method also makes use of the gradient function[41], but it only uses the last layer of the point cloud, which is the distances on the z axis.

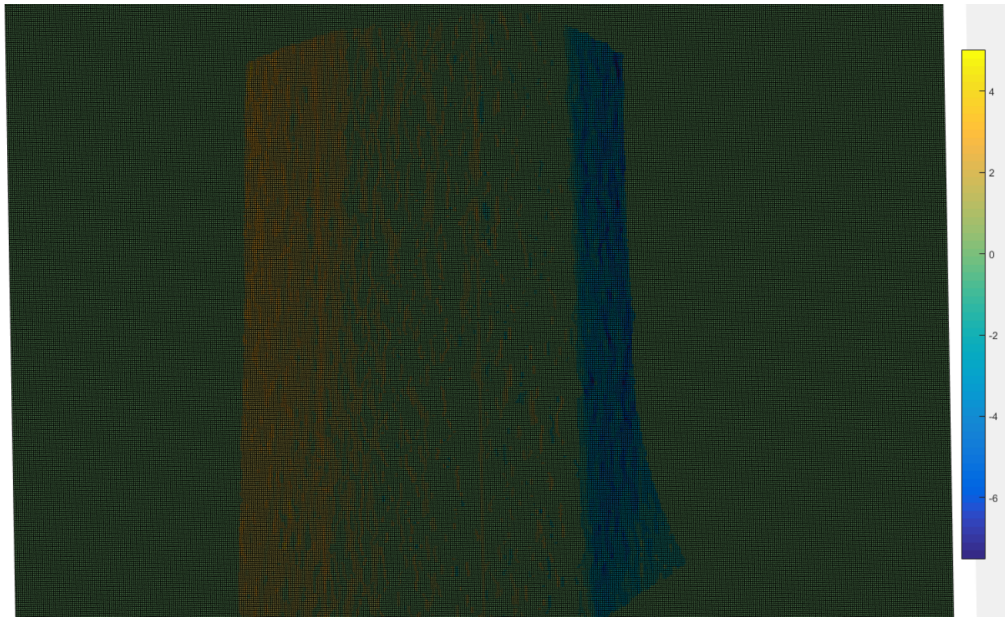


Figure 3.17: Gradient function on a red cardboard

This method first divides the image in equally sized squares. Then, the mean gradient

value[41] of each square is calculated. Five mesh densities are also created. For this tak, the function meshImage accepts as input parameters the objects location and the interval, and returns the mesh on the objects surface.

To assign one of the mesh densities to any square, their gradient mean was saved into a matrix. Then, if the mean is in a certain interval, a number is assign from one to five, which is the same as the mesh densities. The interval only covers mean gradient values from one to bigger than six. The intervals are, less than one, one to two, two to four, four to six, and bigger than six.

Then, with the one to five matrix, it is necessary to return this matrix to the original size, as it has only one number for each square. To achieve this, each point of the matrix is replicated, to create a square with the same dimensions as the original squares.

Finally, the matrix of one to five values with the same dimensions as the original, is divided into five with the same dimensions but one matrix for each density. These matrixes are then multiplied by the respective mesh density and in the end they are added.

The result is an image with white points with different densities, already with the information on the objects location, that came with the creation of multiple densities.

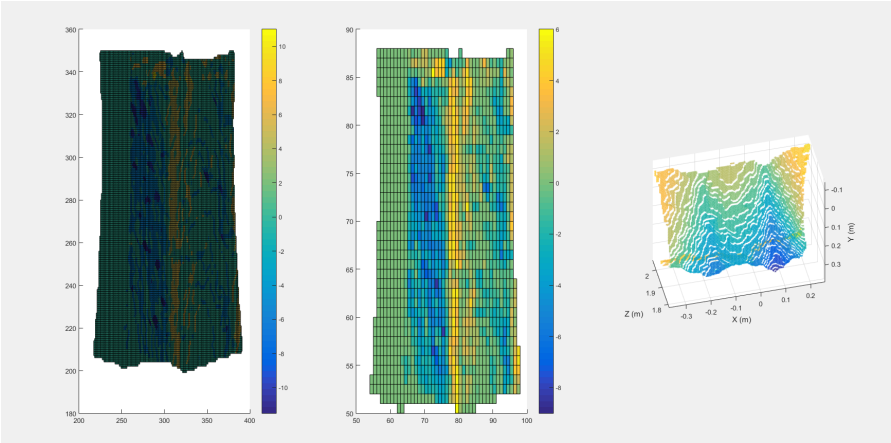


Figure 3.18: Gradient, Division in Squares and Point Cloud

Similar to the other two mesh creation methods, a function was created to allow a cleaner and more structured code. The function has the name getMeshSquares and accepts as input parameters the gradient value on the z axis, which is the sum of the vertical and horizontal gradients, the objects location, the interval between points for the thinner mesh, and the square sizes.

3.6 Outer layer of the mesh

It was observed that it was necessary to refine the mesh in the edged of the object. To perform this task, the canny[43] filter was used. The use of this filter returned an array with the contours of the object. To know the coordinates of these contours, the bwboundaries[95] command was then used. This command returns the contours in a cell form, where each cell has one array of coordinates for each contour.

The contours in the cell object represent the most refined mesh, on the edge of the object. To allow a thicker mesh, a cycle is used to select numbers, for each cell in an equally distanced interval. The result is another cell object but with less coordinates than the first.

Then, after the creation of the boundary points, it is necessary to add them to the already exiting interior mesh. So, another cycle was used to read each number of each cell and set those coordinates as white pixels in the inner mesh image.

A function, with the name getObjectContours, was created to incorporate this sub-chapter code. It accepts as parameters, the objects location image, the inner mesh image, and the interval of selection of the contour points on the cells. The return of this function is the y and x points of the mesh.

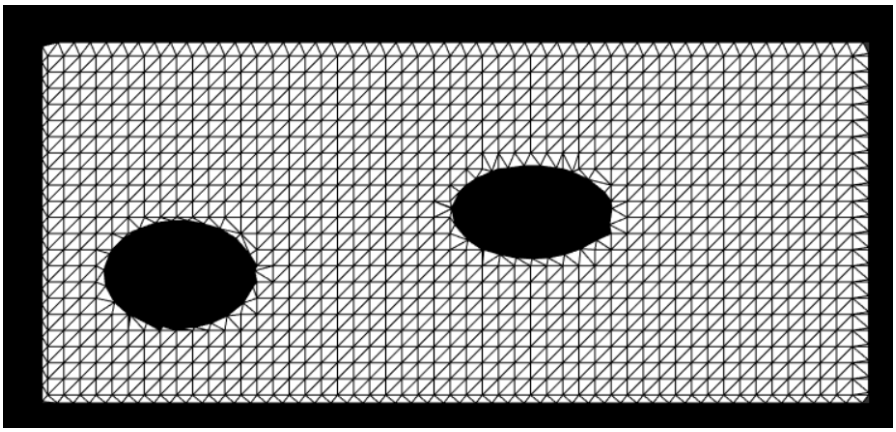


Figure 3.19: 2D mesh with delaunay triangulation

3.7 Final Mesh

The Delaunay triangulation[44] was used to create a triangulation with the previous calculated points as vertices of these triangles, which, in resume creates a detailed mesh on the edges of the object, interior and exterior.

This concludes the vision component of the work.

Chapter 4

Automatic Orientation of the laser beam

This chapter describes the creation and processing of the messages from the Arduino to MATLAB and vice versa, and the programing of the Arduino Uno and its shields.

4.1 Construction of the measurement coordinates

The measurement points are sent to the Arduino using an angular format, which makes easier its programing. The vectorial information from the Delaunay triangulation is available which is then set into a binary image that has its points compared with the point cloud to find the real-world coordinates. The smaller than 0.8 values in the third layer of the coordinates matrix are then set to not a number, and the matrix lines with not a number values are then deleted, to clean undesirable values.

4.1.1 Calibrations of the laser

The real-world coordinates need to be calibrated, as they are defiend using the Kinect data and it is necessary to have them represented by the galvanometer.

As mentioned before, with the prototype box, the distance between the Kinect and the galvanometer is always the same, allowing a calibration with fixed values.

The first thing noticed was that with zero volts, the laser had an angle of 3 degrees on the x axis and 4.5 on the y axis of the galvanometer. This compensation is then automatically added when MATLAB is calculating the angle of each coordinate.

With the beam angle corrected, only translations need to be done and, to calculate its values, x, y and z, the laser was set to the position zero. Then with the Kinect, the distance x and y between the point drawn by the galvanometer and the zero of the camera is calculated by the centroid of the point. The x and y distance are -41.5 mm and -125 mm respectively.

The final coordinate, z was measured with a measuring tape, where it was possible to see that the distance was the same in the Kinect and the galvanometer.

To find the points angles on a galvanometer perspective the translation matrix is then calculated which in resume, corresponds to add the x and y calibration to the Kinect x and y respectively.

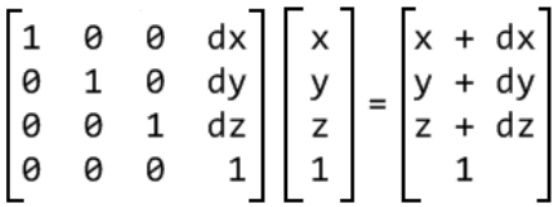


Figure 4.1: Calibrations Matrix[45]

4.1.2 Creation of the string to send the coordinates

In order to avoid unnecessary travels to the galvanometer mirrors, which would be drawing the coordinates in a random way, the matrix, with the coordinates is sorted. First the sorted points are in the x axis, since choosing between y or x the would not make any difference. This operation creates also a sort index, which means that it is possible to order the other two axis, y and z, with the order of the x axis.

The galvanometer works in angles, since it is known that 5v correspond to 10 degrees and 0v correspond to 0 degrees, all the other values are linear. So, with the sorted distance matrix, it is possible to calculate the xz and yz angles between the galvanometer and the measurement points. Using trigonometry, it is known that the tangent of an angle is the ratio of the opposite leg to the adjacent leg of a triangle, so, knowing x and z or y and z, by doing the inverse tangent, atand [46] on MATLAB, of z/x and z/y, the opposite angle of the necessary one can be obtained, but by performing the subtraction 180-90, which represents the sum of the interior angles and the right angle respectively, and subtracting the inverse tangent calculated at this value, the result is the angle between the galvanometer and the measurement point.

After the implementation of this method to calculate the angle, another easier way was considered, that was, instead of the division of z/y or z/x in the calculations just mentioned above, the use of x/z and y/z would avoid the subtraction 180-90. Since both solutions have the same result, the first implemented solution was used.

With the correct angles, a rounding operation was done to only have two decimal cases on the angle. This is necessary to avoid, sending more decimals cases than the required ones. This also keeps the message shorter.

The final step is concatenating both vectors, angle x and angle y, in a matrix. This matrix is also read, and the its numbers are written in a string, using a for cycle.

The message composition has two ";" to separate other information that are written in this fields, to send commands to the Arduino to perform tasks that are not related to the measurement points, as for example to show the limits of the galvanometer or change the interior light color. Then the number of points is written, and another ";" terminates the sentence. Each field in the message is separated with ";". Then, the measurement points are written. First one point x and its y coordinates until the matrix is completely read.

To perform the task from finding the real-world coordinates until the message creation, a function was created. Its input parameters are the point cloud of the measure-

ment points and the translation coordinates x, y and z for the calibrations, and it returns the message to send to the Arduino.

4.1.3 Send the coordinates to the Arduino

With the message created, it is necessary to send it to the Arduino. To perform this task, first it is necessary to connect to the Arduino, setting the baud rate, the output buffer size, the COM port and the terminator.

The baud rate is the speed of signal or symbol changes that occur per second. The value 115200 was chosen because the time it takes to send a message and view its effect on the product was imperceptible, and was also the fastest speed advised by arduino[47]. Then, the output buffer size had to be changed too. It represents the total number of bytes that can be stored in the software output buffer during a write operation and the usual value is 512. The need to change this value was due to an error that occurred as the output buffer could not hold all the data to be written. The COM port, as the name indicates, is the name of the COM on which the Arduino is connected. The final parameter is the terminator. It had to be defined as a way for the MATLAB to know when to stop reading a message from the Arduino.

With the connection parameters defined, the next step is to open this connection. To do this task, the command fopen[48] it is used. This command opens a connection with the Arduino if it is connected. If the Arduino is not connected, this function gives an error.

The next step in sending a message is the use of the command fprintf[49], which sends the message itself. This command accepts two parameters which are the connection and the message. If the Arduino is somehow disconnected and the fprintf command is executed, it also gives an error.

Then, to read the COM in order get any message, the command fgets[50] is used. The only parameter is the COM port.

The last used command on sending a message is the fclose[51], which closes the COM connection. These four commands complete a communication cycle, open, write, listen and close.

To avoid an error by message communication on the code, which would stop it from working, the TRY[52] function was used which allows the program to try to execute the code and in case of an error it executes another or displays a message. This is used in the app that makes the connection from the MATLAB to the Arduino and is discussed on the next chapter.

4.2 Reading the message on the Arduino

With the MATLAB component to send and receive messages complete, it is necessary to translate those messages into movement on the galvanometer.

To communicate with MATLAB, the Arduino must first initialize the serial COM. This is made by giving the serial communication the same baud rate as in the MATLAB, in this case 115200 bits per second, and starting it, using one command, Serial.begin(baudrate)[56].

With communications active, the Arduino needs to send and receive information, which is made using the Serial.read()[57] and Serial.println(text)[58] respectively.

4.3 Sending MATLAB messages to the Arduino

To decode the string sent from MATLAB, the first command tried was readStringUntil(terminator) [60]. This command reads the string until the terminator character is found. By using this command multiple times, it was possible to separate the value of the string to variables. The only limitation was the fact that it had problems reading large strings as for example 10 points to draw, as it loses part of the string.

Once communications are active, every time the Arduino receives something on its com, the serialEvent[59] will be active.

To avoid losing information, the function serialEvent was used to replace the readStringUntil function. Inside this function, a while loop was created to keep the cycle active until the serial communication stopped being available. Inside the while loop, and to read the serial it is necessary to use the Serial.read function which then saved its result on a char variable. Then, that variable is saved in another variable that has the function of recording the entire string. The first char variable is then checked to see if it is an end of line and if so the string is completed.

In resume, this void function reads the string received, character by character and saves them into a string that is a global variable. It also sets a global boolean variable as true, to inform that there is a string to read.

4.4 Split the string in the Arduino

The received string needs to be divided into multiple variables, and as they are separated by an ";", the division looks for this character as the separator.

With the received string from MATLAB saved in a variable, which is named InputString, to select a value between two separators, it is necessary to know what is the position of those two separators. With the task of extracting a portion of string between two separators, getValue function, from stackoverflow[61], was used. It accepts as input parameters the string, the separator and the position. What the function does is creating a counter that saves the position of the previous terminator and goes looking for the next. If it finds it, it saves it as last position, otherwise it saves the last character position of the string. To know if a character is the terminator, the function reads the string, with a for cycle, until it gets a value on the separators counter bigger than the position asked.

The function returns the string between those two positions and, in case of the position asked being bigger than the number of separators, the output of the function is an empty string otherwise it returns the string between those two separators.

By being able to know the value between a certain terminator interval, a for cycle is used to read the string, until it reads all the galvanometer positions. The number of positions is also obtained with the getValue function, which input parameter is the third position.

4.5 Controlling the galvanometer

The control of the galvanometer is performed using functions created for this effect. To move the galvanometer on both axis, four functions are used. One for the positive movement and another for the negative movement along the axis of both axis.

These functions are named, "moveyneg", "moveypos", "movexneg" and "movexpos", and accept as input parameters the angle to draw and are called by another function "Desenhar_coordenadas". This function accepts as parameters the x and y angles and is called by another function named "Ler_vetor" without input. The "Ler_vetor" function is then called on the for cycle if there is a string to read .

The division of the control of the galvanometer in multiple functions is due to the fact of structuring the code, making it simpler to understand.

The "Ler_vetor", reads the string received from MATLAB, starting at the third position on the function that splits the string, and continues until the number of lines is reached, which is the second position using the split string function, "getValue". This task is performed with a for cycle. Inside the for cycle, the string is splited with "getValue" to extract the code in a certain position. Then that valued is converted from string into float. Float allows the number to be more precise than integer, which has just natural numbers, but it requires more space. This first number is the x angle. Then a counter is added to get the next position, which will be the y angle. The same treatment used for the x angle is used for the y angle. Then, with both angles, the "Desenhar_coordenadas(x angle,y angle)" function is called. After this function, a pause is used to allow the mirrors to reach the intended position, where the laser pointer is then turned on, waits a few microseconds and is then turned off again, and it completes the cycle, which repeats until the number of lines is reached.

Inside the "Desenhar_coordenadas" function, the angle is checked to see if it is bigger or smaller than 10 or -10 degrees respectively, and, if so, the angle is set to 10 or -10 respectively. Then, with the points inside the limits, the angle is transformed from 0 to 10 degrees into 0 to 255 value. The 255 value corresponds to 5V on the Arduino as the 0 corresponds to 0V, and, in the galvanometer characteristics datasheet, it is mencioned that, with 5V the galvanometer is at 10 degrees. This transformation is made using a cross-multiplication, which gives, with two operations the two angles in the scale 0-255. With this value, the move functions are called using an if then else condition, for both axis. If the x angle is bigger or equal to 0 the "movexpos(angle x)" is called, otherwise if the angle is less than 0 the "movexneg(angle x)" is called. The same is done for the y axis. And this completes the "Desenhar_coordenadas" function. The move x and move y functions are the ones that output values on the Arduino pins, but before get into them, it is necessary to understand how the Arduino movements are controlled.

The galvanometer needs an analog signal to be controlled and the analogWrite[62] function on the Arduino produces a signal that simulates an analog signal, using PWM. The PWM signal is only available on 6 Arduino pins, and two of them, the fifth and sixth pins, control the delay functions, which means that, if changed, the value set on the delay function changes, so, for the interest of the work, they are not changed.

4.5.1 Pulse Width Modulation

Pulse Width Modulation (PWM) is a modulation technique used to encode a message into a pulsing signal but is mainly used to allow the control of the power supplied electrical devices[63]. The pulsing signal, which is turning the signal on and off, by having a higher frequency than the one that the device can read, it is possible to, with a digital signal, simulate an analog signal. The percentage of time the digital signal is on, is the duty cycle. So, with a duty cycle of 50% on, the output voltage would be around 2,5V.

This method is only available on six of the Arduino pins, which are controlled with a timer. Timer 0 controls the 5 and 6 pins, timer 1 controls 9 and 10 and timer 2 controls pins 3 and 11 for the Arduino UNO.

The use of a low pass filter, which consists in adding a resistor and a capacitator, in a PWM circuit, allows to stabilize the signal curve by charging the capacitator when the digital signal is on and discharging it when its off creating a stabilized DC voltage signal proportional to the duty cycle. This way, the use of a small charge capacitator is advised.

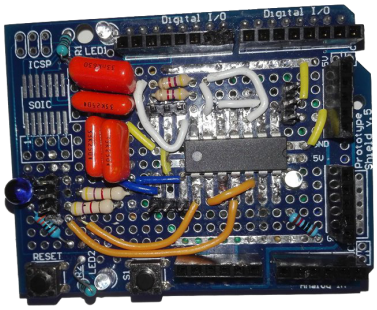


Figure 4.2: PWM Shield

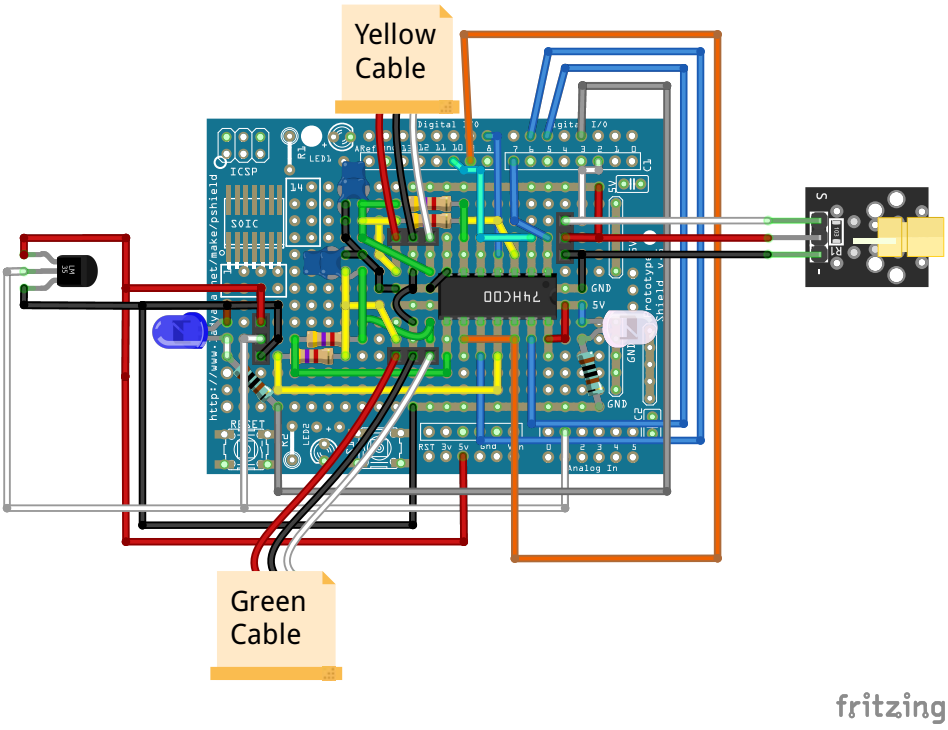


Figure 4.3: PWM Shield Schematics

In Figure 4.2, it is visible that the shield, along with the low pass filter, allows the direct connection of a laser pointer and a temperature sensor.

The PWM shield has a logic gate NAND. This logic gate was used to reduce the number of low pass filters, as only two PWM signals would be necessary, one for x and one for y, so the logic gate had the primary function of accepting a PWM signal, already filtered on the low pass filter, and sending it to the positive or the negative wire of the galvanometer axis. With four gates, the control of each gate would be done by an Arduino pin as two gates are used for each axis. The signal of each axis gate group comes from an Arduino PWM pin, in this case the ninth and tenth pins. The axis control is done by pins 7 and 8 for the x axis, positive and negative movement respectively and by pins 5 and 6 for the y axis, also positive and negative respectively.

A logic gate is basically a boolean operation where two digital signals return a logical signal which depends on the value of the first ones. Digital signals have two positions, on and off which are 1 and 0. The NAND logical gate negates the signal that exist the gate, for example if both signals are 1 the result will be zero. In case of any or both signals being 0 the result will be one. With this information, it was expected that by sending 0 on the digital gate would result in a 1 with any analog signal value and by sending 1, it would result in the analog signal negation which means that instead of a duty cycle of 80%, it would be 20%.

By connecting the logic gate in the previously described way, the Motorola's 74HC00 gate worked well in some gates, as it has four, but, by failing at one gate, this method did not worked. So, to check if it was not a logic gate error, a Philips AND gate 74HC08 was applied but the result was worse, as none of the gates provided the desired effect. The final step was to send a digital PWM signal to the gate and then use a low pass filter before sending it to the galvanometer, being this the only way for it to work. With this conclusion, the two low pass filters solution had to be changed to a four low pass filter solution where the logic gate has the task of avoiding the use of more than two PWM pins.

For the management of the logic gates of the Motorola 74HC00, the "`move x`" and "`move y`" functions mentioned before were used. To move the galvanometer mirror in the x positive axis direction, the function "`movexpos`" which means "move x positive", will first see if the last position was on the x positive axis, if not it calls a function called "`xpositive`", which then sets the x positive high and the x negative low.

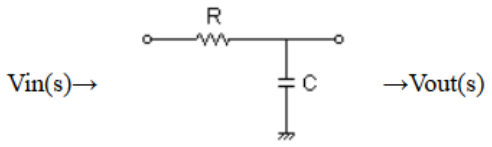
Table 4.1: Arduino movement functions

Function Name	Pin High	Pin LOW
xpositive()	7	8
xnegative()	8	7
ypositive()	5	6
ynegative()	6	5

Table 4.2: How the movement functions are called

Function Name	Functions Called
movexpos()	xpositivo()
movexneg()	xnegativo()
moveypos()	ypositivo()
moveyneg()	ynegativo()

The low pass filter solution is calculated according to the PWM frequency, which then sets the resistors and capacitor values. Since the galvanometer draws 20 000 points per second, the Arduino PWM frequency should be higher than that. Then, as the capacitors should have a low value, the resistors value needs to be calculated with these two constraints. The capacitors used have a 10nF capacity and, to have a higher than 20kHz frequency, the resistors had to have less than 8.2kΩ . As the only available resistors have 4.3kΩ, the frequency had to be set to 33862.75 Hz.



$$fc = \frac{1}{2\pi RC}$$

Figure 4.4: Frequency calculations for a low pass filter

Other way to control the galvanometer with PWM is the use of built in functions of the Arduino, which are explained in Appendix C.

4.5.2 Galvanometer control with DACs

The second solution to obtain an analog signal uses a DAC, which means digital to analog converter. The DACs used are the MCP4725.

The DAC uses Inte-integrated Circuit (I2C) Protocol, to comunicate which is a protocol intended to allow multiple slave devices to communicate with one or more master devices.[66]

The MCP4725 has six pins, the OUT which is the output analog signal, two GND signals which are the ground connections, the VCC which is the 5V power supply for the DAC, and then the I2C pins, the SCL and SDA. The SDA pin, which means Serial Data allows bi-directional communication with the master, by bi-directional its meant that the input and output communications use the same pin. As for the SCL pins, which means Serial Clock, preforms the timing between the devices, so the information could be trusted. Just like the low pass filter solution, where three pins are required to one galvanometer axis, this solution also needs six pins to control the two-galvanometer axis, one for each DAC and the SDA and SCL. The biggest difference is that the DAC are connected in serie.

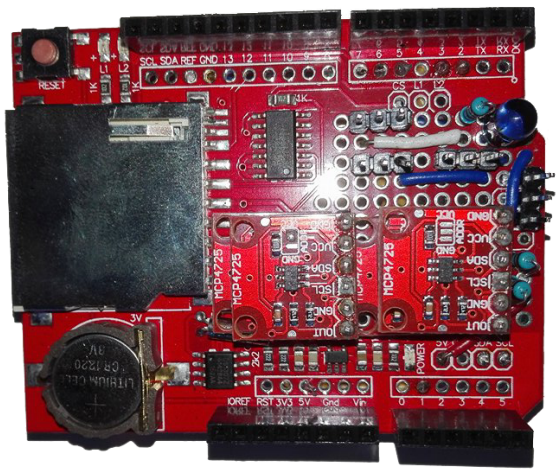


Figure 4.5: DACS Shield

To allow a better understanding of the connections, a scheme was also created,Figure 4.6, where the DACs were placed outside the shield to allow a better understanding of the connections.

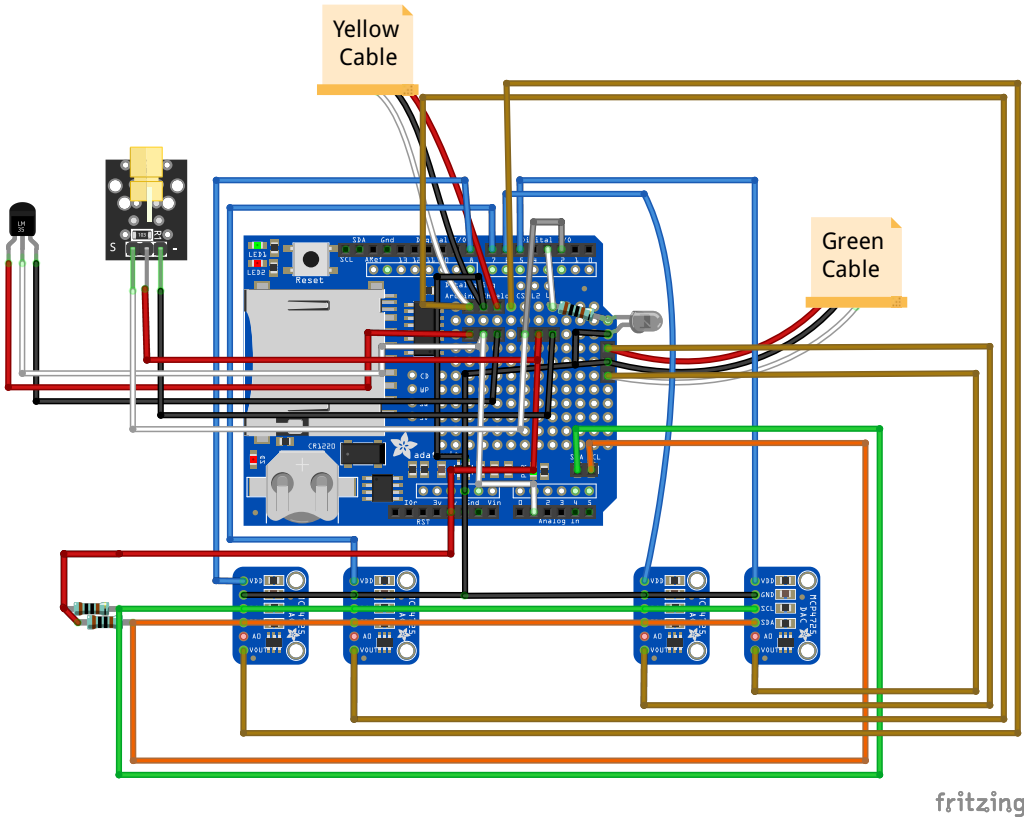


Figure 4.6: DACS Shield Schematics

Both green and yellow cables tags in the scheme are related to the galvanometer

connections, where the green is for the y axis and the yellow is for the x axis. The DAC shield has the same functions as the low-pass-filter shield, and although it has a memory card slot, it was not used on this work. The DAC shield has the same connection pins as the low-pass-filter.

4.5.3 Operate the DAC with an Arduino

To use the DAC shield in the Arduino, just as the PWM library for the low pass filter, two more libraries were used to control the DAC, the Wire and the `Adafruint_MCP4725` libraries[67]. The Wire library[68] is only called to be used on the `Adafruint_MCP4725` library, and it controls the I2C connections, as for the Adafruit library, its goal is to simplify the writing of the code in the Arduino. The libraries are called with the following code:

- # include <Wire.h>
- # include <`Adafruint_MCP4725`.h>

Within the arduino code and before the setup function, the dac function was called with "`Adafruint_MCP4725` dac". What the previous command does is defining the properties of the dac mcp4725 to the variable dac. Then, variable to carrier the dac values was created, "`dac_value`", which is unsigned log 32, since the integer option could not get to the 4095 value that corresponds to the 5V in the dacs. The unsigned long 32 allows can reach values up to 4 billion, and the unsigned long 16 can reach 65.535. But, when the 4095 value is multiplied in cross multiplication, the 16 unsigned long maximum value exceedes forcing the use of the unsigned long 32.

With the libraries imported and the variables set, the next step is to send information to the dacs. Since the I2C wires are in series, the master needs to know to which of the slaves it is sending information, being this distinction done using the address of each DAC. To find the address, an I2C scanner from Arduino was used but only one address appeared, even with all four dacs connected, as result of they being identical copies. A solution to change the dacs address to individual ones, was researched but the conclusion was that it is not possible, as its needs to be done on the hardware. The only possible way is to weld two connectors in order to have two different addresses, Figure4.7.

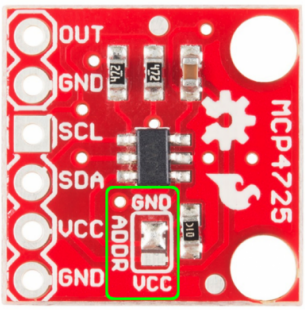


Figure 4.7: DACS Shield 0x62 address[69]

With the first two connectors welded, the address is 0x62 and with the last two welded the address is 0x63. This way the 0x63 address dacs are use for the y axis and the 0x62

are used for the x axis. The control of the dacs, as represented on the dac shield scheme, it performed using the vcc pin, applying 5v or 0v to use it or close it respectively, as the most important note on this way of controlling is that a 5V connection needs to be wired to the SCL pin and to the SDA pin with a resistor, as represented in the scheme, in order for it to work, otherwise the program will crash.

So, with two addresses for each axis, the selection of the axis is done using the dac.begin(address) on the Arduino, and after it is selected, it is necessary to give it a value. For this task, a function setdac(value) was created to do cross multiplication to find the equivalent of a 0 to 4095 in a 0 to 255 scale. Then to send the value to the dac, the function "dac.setVoltage(`dac_value`, false)" was used, where the false means that the value is not saved in the device. If there is a need to save it, it is necessary to change this value to true. The functions dac.begin and "setdac(value)" are called within the move y/x pos/neg functions.

Table 4.3: Dacs address and Vcc pins

Axis	Address	Pin
X Positive	0x62	7
X Negative		8
Y Positive	0x63	5
Y Negative		6

For both methods to work with an easy change of shield, the functions of both methods are executed at the same time even though only one shield is connected.

Chapter 5

Graphical User Interface

One of the objectives of the project is that someone with no knowledge on coding could use the code and procedures of Chapter 3.

For this, it is necessary a user-to-code connection, which is the GUI developed.

The GUI creation process resulted in two versions, one was built when there was no prototype, to validate the vision component, and the second after its construction.

This chapters discusses the GUI, and its integration with the Chapter 3 functions that each button activates.

5.1 GUIDE

The creation of the GUI was done using MATLAB's GUI creation software, GUIDE. This allows for a complete compatibility between the GUI and the functions mentioned on Chapter 3.

GUIDE is the program to develop graphic interfaces on MATLAB. It integrates the two primary tasks of application building, one being the creation and placing of the visual components and the other being the programming of the application behavior. This program allows a quick change between visual design in the canvas and the code development in an integrated version of the MATLAB Editor.

The code development section of a GUI on MATLAB works based on functions, having each function its variables. To send information over functions, it is necessary to create global variables which are defined by writing global before their name. Within each function, to use the global variable defined elsewhere it is necessary to call them using the command "global" and then the name of the variable, otherwise it will not assume that the variable is global.

To display figures, GUIDE uses axis, where, instead of a graphic, an image is shown.

5.2 First version of the GUI

The first version of the GUI, as mentioned before, was created before any prototype being developed. The only parts created until its creation were the bases of the laser pointer and galvanometer, and other two bases to secure the vibrometer.

With these components it was necessary a calibration between the galvanometer and the Kinect since the distance would be different each time the components were mounted,

being necessary a button for this task. Also, to have a simple and intuitive GUI, all the buttons and windows are in the same main window.

The GUI also has a step by step philosophy, showing the different menus when necessary, and if the user clicks on thee first step, for example, it will hide the menus from the third step and show the first and second steps, avoiding errors and forcing the user to follow a specific procedure.

Figure 5.1: First version of the GUI

Figure 5.1 shows the first version of the GUI, where all menus are visible. The mentioned steps start with the camera input panel, which is the only one visible when the application starts. To proceed to the next step, it is necessary to click on the button refresh. Then the axis 3 and 4 appear, as well as the "Edit object properties" panel, which is the second step, and to proceed to the third step, it is necessary to click on the "Confirm Object" button, which shows the "Edit Mesh properties", the "Calibrations" panel and the axis 5. The "Connect to arduino" and "Send Mesh" buttons only appear when the number of exterior points is defined on the "Exterior Points" slider.

5.2.1 GUI step 1 Kinects Image

In the "Camera Input" panel, the axis 1 and axis 2 correspond to the Kinects RGB and depth cams respectively. The RGB camera shows the color video feed from the Kinect, allowing the user to easily place the part for measurements on the angle of view of the camera. The depth cam, shows a color representation of the distances but, due to the shape of the objects to measure, no significate changes will be seen, instead, axis 2 will basically show the same color with small/imperceptible variations, but its use was required to know when the depth cam was producing valid results, so if the "Refresh" button is pressed when the Kinect is still producing the depth image, the results will be invalid, showing something that does not correspond to reality. The axis 1 and axis 2 are used the preview of the videoinput function of both cameras.

There is another feature on this panel that is the "Position" slider which controls the angle of the Kinect in relation to the gravity. This slider has the function of allowing an axis x rotation to position the Kinect in relation to the galvanometer with the same x angle. This angle was set to 0 every time the GUI starts, and then a rotation from 0 to 27 degrees is allowed.

In case of any connection problems, instead of the Kinect video on the axis 1 and 2, an icon appears, signaling a problem. If the Kinect was not detected when the GUI first started, but it was already connected when MATLAB was opened, a click on the "Refresh" button, instead of going to step 2, restarts the application and it might start working, otherwise, if the Kinect USB is connected with MATLAB already opened, the restart button performs the same restart task but the problem will persist, being necessary a restart of the MATLAB program. This problem is due to compatibility issues between MATLAB and the Kinect as sometimes, after its USB plug being connected to the computer with MATLAB opened, works.

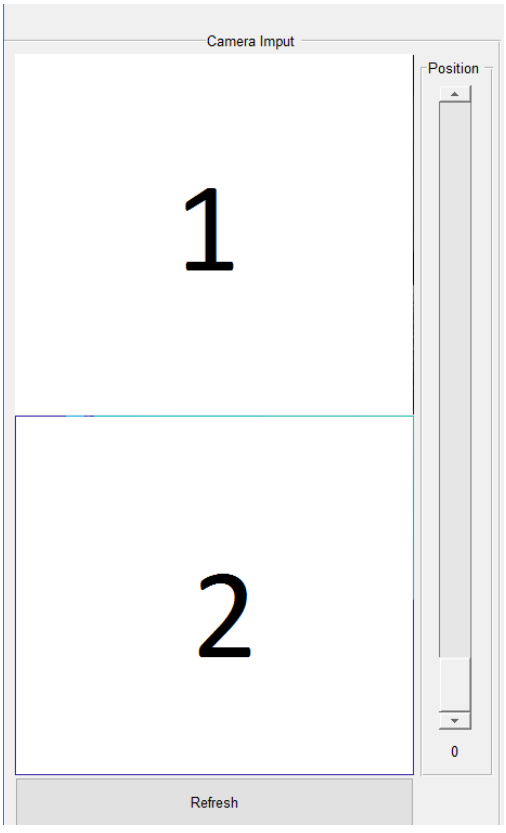


Figure 5.2: Camera Input

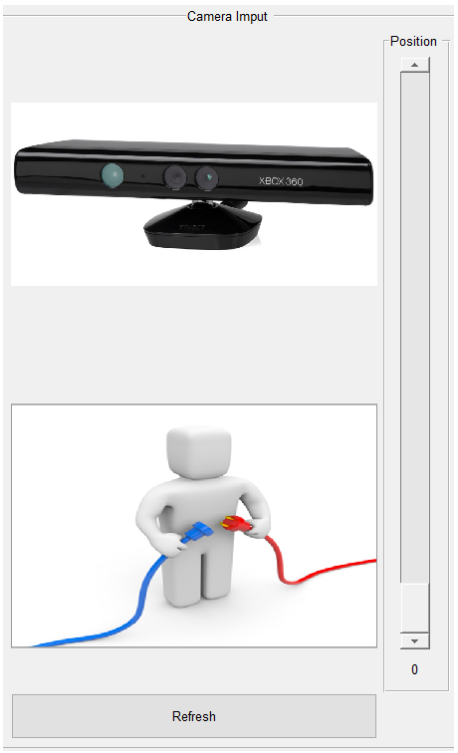


Figure 5.3: Error of no Kinnect connection

5.2.2 GUI step 2 Edit object properties

When the refresh button of the previous step is pressed, snapshots are taken on both cameras, one in the RGB and twenty-one in the depth cam.

These snapshots are then threated to remove the background, and the gradient of the point cloud is calculated. The gradient appears on the axis 4 as the objects location appears on the axis 3. The objects location, as mentioned on Chapter 3, is calculated

using the getObject function, where the input parameter threshold is defined by the slider on the GUI with the same name, which scale goes from 0 to 1. Then, two other sliders can be found, the "Number of Erodes" and the "Number of Dilates". These sliders are rarely used but are intended to, in case of lack of light, allowing an easier removal of the background, as sometimes it can be done using an erode or a dilate. This was the major argument for its use instead of open and close operations.

The dilate operation, adds one pixel along the border of the contour of the objects location and the erode performs the opposite operation.

Figure 5.4: Object Pannel

The object on axis 3 appears with the real-world colors as the logical image with its location is converted into a 3D matrix and then multiplied by the original snapshot, giving it the same values as the original snapshot on the locations of the 3D "logical" matrix.

If by any chance, the erode and dilate sliders are used at the same time, the erode operation is the first one to take place, followed by the dilate, which preforms the same as an open calculation.

5.2.3 GUI step 3 Mesh Creation

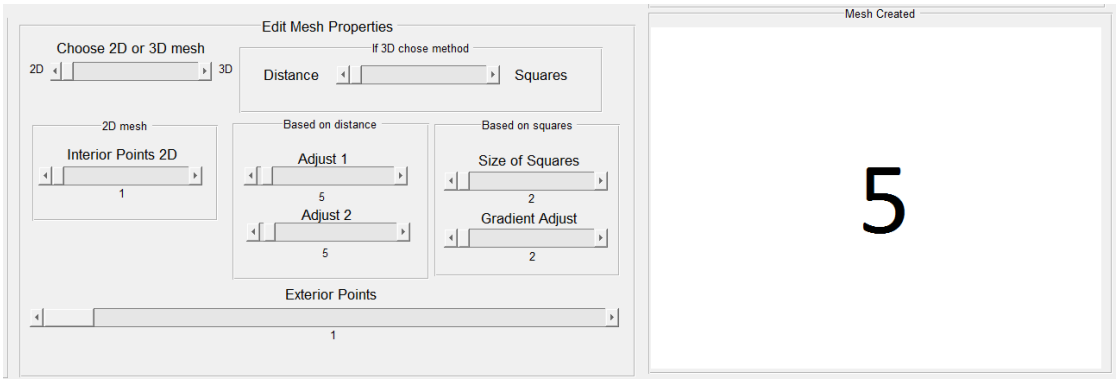


Figure 5.5: Edit Mesh Properties panel

This is the final step of the image treatment functions. As mentioned on Chapter 3, there is a 2D and 3D methods to create a mesh, which can be selected on the "Choose

2D or 3D mesh" slider. The mesh points, calculated using the Delaunay triangulation appear inside the axis 5.

The interior points density using the 2D method, can be choosen using the "Interior Points 2D" slider on the "2D mesh" panel, having an interval that goes from 1 to 50, which is also the pixel distance to each other. As for the 3D methods, first is necessary to put the "Choose 2D or 3D mesh" slider on the 3D side, otherwise no changes will appear in axis 5. The next step is to choose the distance or the squares method, which can be chosen on the "If 3D choose method" panel, by pulling the slider to the wanted side. If the division based on the gradient distance is selected, the changes in the "Adjust 1", which has an interval from 2 to 100 that corresponds to the distance of the points to each other, and the changes on the second adjustment "Adjust 2" slider, which also has an interval from 2 to 100 and corresponds to the selection of only a certain number of points of the adjust 1 distance, will appear in the axis 5.

The other alternative is the medium gradient calculation of the square divided image, where the size of the squares in pixels is changed using the "Size of Squares" sider on "Based on squares" panel which has an interval from 2 to 20, and then the gradient is adjusted by multiplying the number on the slide "Gradient Adjust" by the predefined gradient intervals mentioned on chapter 3, having an interval from 1 to 10.

The last thing to do for the mesh creation process, is the number of points in the contours, which is defined by the "Exterior Points" slider that has an interval from 1 to 20, and performs the canny filter points selection.

These sliders work with invisible buttons, as everytime the exterior points are changed, the interior points are kept. But, if any of the values of any interior point sliders is changed, the exterior points are calculated again as it is the exterior points slider that updates the axis 5 image, and makes the Arduino connection buttons appear, since without points to send, an active button would not be used.

5.2.4 Mesh Calibrations

Since there was no prototype at that time, there was the need to perform calibrations between the Kinect and the galvanometer every time the application was started.

In the "Calibrations" panel a "Calibrate check button", when checked, every time the calibration slider is moved, it forces the upload of the mesh coordinates to the Arduino, that needs to be connected first.

The process of calibration consists in a sum or subtraction on the three axis, x, y and z, which values can be set on the sliders on the "Calibrations" panel. These calibrations are in centimeters but, since the values are not integers, it is possible to set them in a 0.1 mm scale.

5.2.5 Arduino connections

The connection to the Arduino is done using the "Connect to Arduino" button, that when clicked and if the Arduino is connected, it shows a "Connected to Arduino" message on the messages panel. If the Arduino is not connected, it shows a message saying that the Arduino is disconnected or in another COM port. This version of the GUI did not accept other COM ports besides COM3, which is the standard for the Arduino Uno.

Figure 5.6: Arduino connection panel

With the Arduino Uno connected, the "send mesh" button is visible, allowing the user to send the coordinates of axis 5 to the Arduino.

5.2.6 Overview of the GUI for vision valitadion

With all the steps active, except for the Arduino connection, a global view of the discussed problems and results of the GUI can be done. In the axis 2 is imperceptible to distinguish the blue carpet on the depth image, and the gradient image has the same color since the gradient between each point is very small, which was also expected knowing that even with some inclination on the Kinect, the carpet was parallel to the floor, resulting in the same gradient along the object.

Lastly, although this GUI version is made to work with white parts on a black background, it can capture any other objects provided that the background has a big contrast against the object.

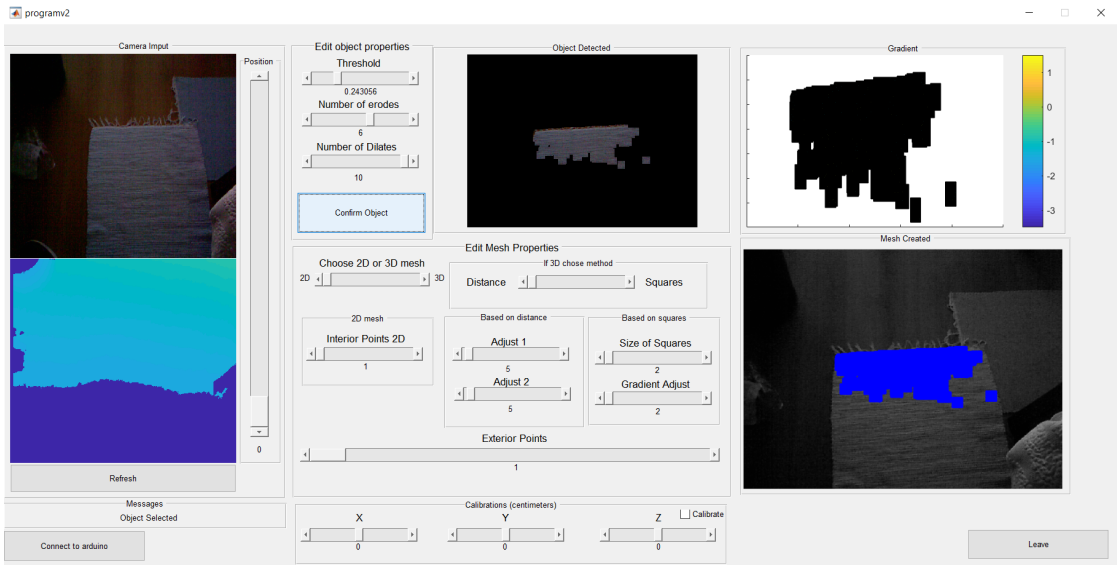


Figure 5.7: Test with the GUI

5.3 Final version of the GUI and arduino code

With a working prototype, changes were done to the previous application. First it needs to be able to read the temperature input from the sensor on the Arduino, secondly it

had to be able to control the IR lights in the box, thirdly the calibrations value was on the calibration slider, fourthly a list-box was added were it was possible to select any available COM, and finally a point cloud button was added that switches between the depth cam and the point cloud.

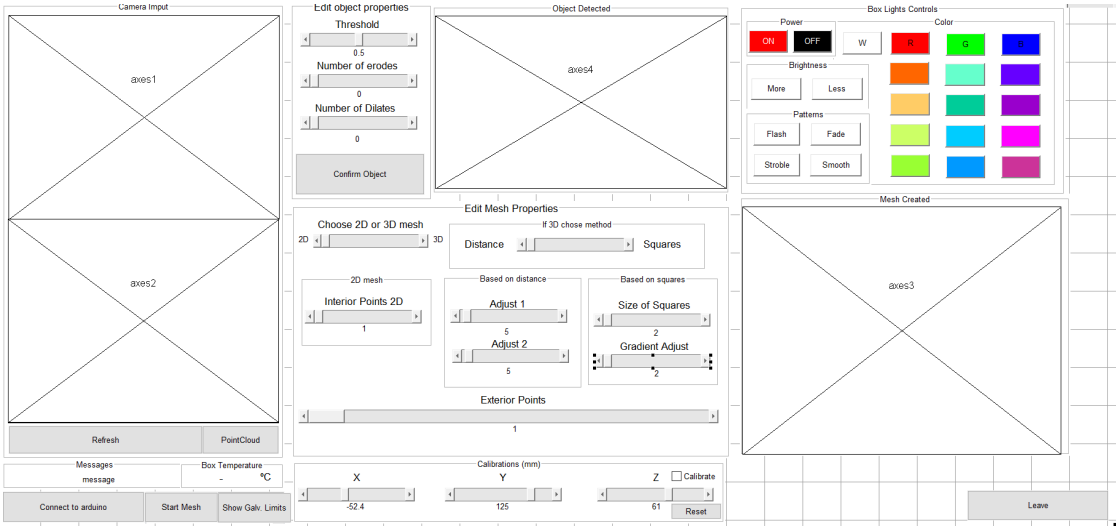


Figure 5.8: Final GUI

5.3.1 Temperature readings on MATLAB

Starting at the temperature reading, a message box was added to show the temperature in Celcius degrees inside the box of the prototype. The temperature can only be seen when the Arduino is connected, by pressing the button "Connect to Arduino". Since the temperature needs to be read in a timed interval and MATLAB does not work in a loop mode unlike an Arduino, a timer had to be used to perform this task.

GUIDE creates a function with the name `"ProgramName_OpeningFcn"`, where "ProgramName" is the actual name of the program. This is the function that is called when the program is started, and the place where the timer was inserted. The MATLAB timers are created as handles, where the execution mode needs to be set as fixed rate with a certain period. It is then necessary to assign the timer to the address of a function so every time it runs, the function is called. The timer function code for the temperature readings is:

- `"handles.timeAPP=timer('ExecutionMode','fixedrate','Period',0.001 ,'TimerFcn',{@uptade_app,handles});"`

The function of the temperature is the `"update_app"` function and the timer was named timerAPP, since it was the first timer of the app.

The next step is to turn on the timer, and it can not be done on the opening function, since the program would crash, so the timer is only started when the "Connect to arduino" button is pressed. To start the timer, it is necessary to call, "start(handles.timerAPP)". [94]

- "a=serial(com,'Baudrate',115200,'OutputBufferSize',
  1500,'Terminator','$');"
- "a.BytesAvailableFcnMode='terminator';"
- "a.BytesAvailableFcn=@uptade temperature;"

Within the temperature function, "fgets" is used to read the COM and, as the only message that the Arduino sends is the temperature, the received message is saved in a variable that is read in the update application timer just mentioned.

5.3.2 Temperature readings in the Arduino

Since the temperature updates does not need to be done with the same frequency of the drawing of the points, a timer was also created in the Arduino to only send temperature updates every second. To create a timer in the Arduino, the library timer was added using a # include "Timer.h", and then a timer variable "t" was created, "Timer t;". With the timer named "t", the information on the frequency of its clock had to be introduced within the setup routine. A tick event was then created to repeat every second. Within the loop cycle, the timer is then updated. In the following figure, the variable "Repeats" is the function that reads the analog pin 1, calculates the temperature and writes that information on the COM port.

- "int tickEvent=t.every(1000,Repeats);"
- "t.update();"

The temperature updates are calculated with the following formula as indicated by its manual.

- "temperature = (5.0 * analogRead(1) * 100.0) / 1023.0;"

5.3.3 Serial port detection

To connect the Arduino to MATLAB using serial communication, the COM port is usually the COM3, but in some cases it can change. The first version of the GUI could not change COM so, if the Arduino from the prototype was not connected on COM3, there would be no way to correct the problem. To solve this problem, the new GUI can detect which COM ports are available and display them on a popup menu. The first value in the popup menu is the selected COM for the connection. To change it, it is only necessary to open the popup menu and select another port, if available.

Then, an "Auto-Refresh COM" button is available, that, if clicked, starts a timer like the temperature timer just mentioned. This button is only to activate or deactivate the automatic detection of the coms . It was necessary to put this function as a button, instead of starting the timer when the program started for the same reason as the temperature timer, the crash of the GUI. To refresh the popup menu when the program starts, the GUI first looks for any available COM and writes it/then on the popup menu.

To automatically detect the serial COM ports available, the following code was used.

- "serialInfo=instrhwinfo('serial');"
- "COMS=serialInfo.SerialPorts;"

First, all the com ports info is searched, then the serial ports found are saved in a variable under cell format. If this variable is empty, there are no available COMs and the popup menu text is "NO COMS AVAILABLE", otherwise, the COM port names are written.

5.3.4 Point Cloud representation

The depth cam on axis 2 of the GUI, as mentioned before is used to allow the user to know when is the depth cam of the Kinect working properly since it takes some time to start. With the depth cam, it is hard to see if the object is being detected or not, so, a solution that allows a change from the depth image to a RGB point cloud representation was added where a snapshot is taken from each cam, and the point cloud is produced.

To change from point cloud to depth cam and vice versa, a button was added to start/stop the timer. The timer refreshes the cloud every 0.25 seconds, since if it was smaller it would result in a crash of the GUI and bigger would not produce the desirable effect, which was a "real-time" point cloud. Also, the point cloud code on MATLAB takes close to 0.25 seconds to calculate, which slows down the application, so this resource should only be used to view the point cloud to check if the object is being detected and then, the timer should be stopped.

5.3.5 Show the galvanometer limits

In order for the user to know where to place the object so it could be on the drawing area of the galvanometer, a new functionality was added to the new GUI, the button "Show Galv. Limits". This button sends a message to the Arduino to turn on or off this functionally that is a for cycle on the Arduino with 40 points that draws a square on the edges of the galvanometer limits. The message to turn on is ";1;" and to turn off is ";2;".

Chapter 6

Results

This chapter presents the results from the first steps of the project until the final prototype. A discussion of the results with the different shields is also presented.

6.1 Software Results

6.1.1 Triangulation Results

The final look of the mesh is created using the Delaunay triangulation as mentioned on Chapter 3.6. Using this triangulation method, a virtual tolerance was created between the contour of the object and the mesh points, since the points used for the mesh are centroids of the mesh triangles.

This meshing method however, since it selects the centroid of the triangles, and, as they sometimes are close but not aligned with each other, the result appears as random points, even though the triangular mesh looks organize.

In certain occasions where the triangular mesh has triangles aligned with each other, if the triangles height is bigger than the width, the centroids of the triangles will not be aligned, so the triangulation transforms a previously aligned mesh into a double line of points, as Figure 6.1 tries to illustrate.

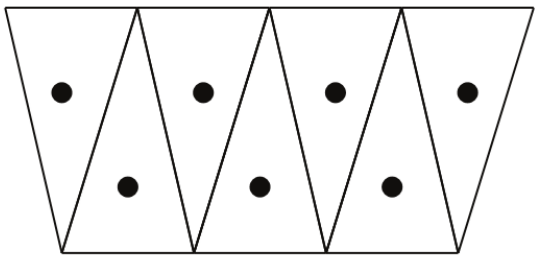


Figure 6.1: Illustrative drawing of the mesh

Figure 6.1 is an illustration of the previously described situation, where the triangles are aligned but the result would be two lines of points closed to each other, which could not be the desired result.

The Delaunay triangulation is created using two types of mesh, the edges and interior meshes. As the exterior mesh is thinner on the internal contours, it adds more measurement points closer to the interior holes, which, may not be the intended result.

This meshing method, on the other hand, allows for a smooth integration between the exterior and interior measurement points. In the 3D interior meshing methods, it creates smooth transitions between the mesh points, and, since the mesh has a finite element look, the results could possibly be integrated into future calculations of this kind using the vibrometer results.

6.1.2 Canny Filter and bwboundaries functions

In the first trials creating a mesh, only the interior points were used. Since the mesh density was equal along the object as the first methods tried were 2D, the result presented some issues. The mesh had problems on the edges, both interior and exterior, as the triangles most of the times were not constrained by the part limits, creating triangles which vertices were in opposite sides on an interior hole. Besides, the creation of triangles inside holes, on the edges, the mesh was very coarse, having some spaces without triangles and others with too many.

By using the canny filter, the inside mesh points had contour points to which they could connect to make triangles. The canny filter detects the edges of the object in the logical image whith the objects location.

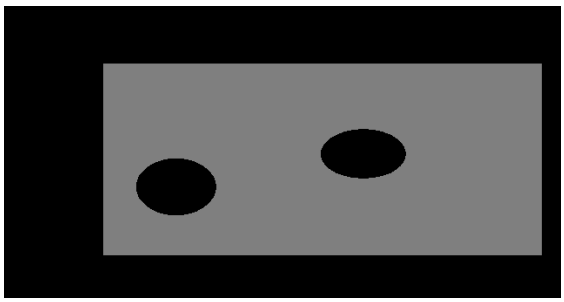


Figure 6.2: Original Object

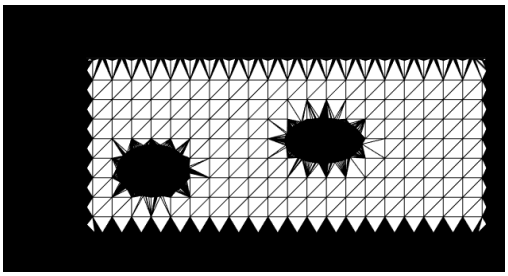


Figure 6.3: Delaunay triangulation

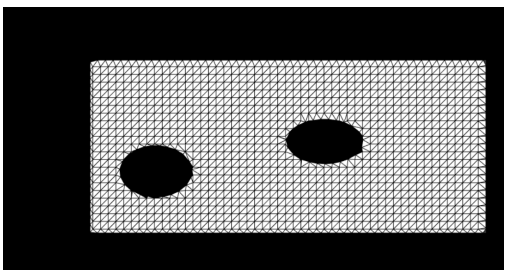


Figure 6.4: Triangulation with canny filter

As it can be seen in Figures 6.2 to 6.4, the introduction of the filter, creates a smother mesh. The mesh can also be more refined on the edges than in the interior, creating a finite elements mesh look, and since the edges of the part are the usual place for the

appearance of cracks and failures, they should also be the ones where more processing power should be applied.

6.2 3D Mesh results

The interior mesh creation for the 3D methods was based on four objects, besides de red cardboard. The expected result should create a more refined mesh along high gradient zones and a less refined on the others. At this stage, none of the objects had paint, being a white paint cover applied after these results. At the time of this results the objects were wrapped in paint paper where some of the edges had no paper and as a result, no mesh.

As discussed on Chapter 3, two 3D methods of mesh creation were created. To facilitate their distinction, the method described in Chapter 3.5.1 is named distance method and the method of Chapter 3.5.2 is named squares method.

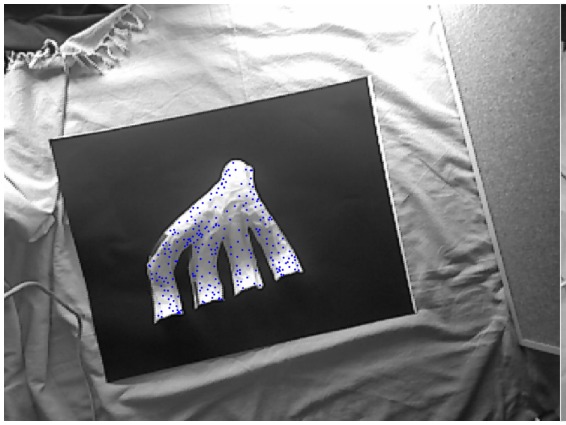


Figure 6.5: Mesh with distance method

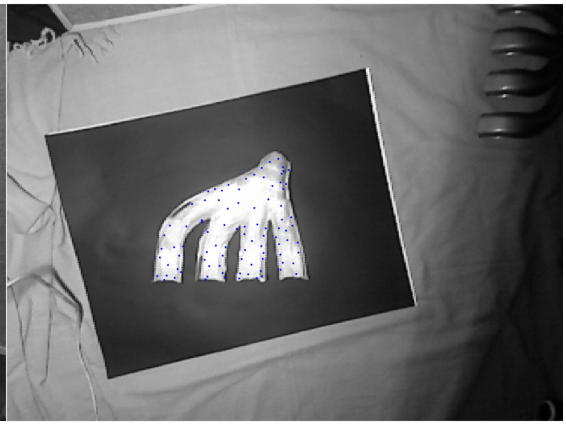


Figure 6.6: Mesh with the squares method

Figures 6.5 and 6.6 the mesh results using an object that represents what the prototype should encounter in the daily use, which is a regular surface with a smoth gradient.

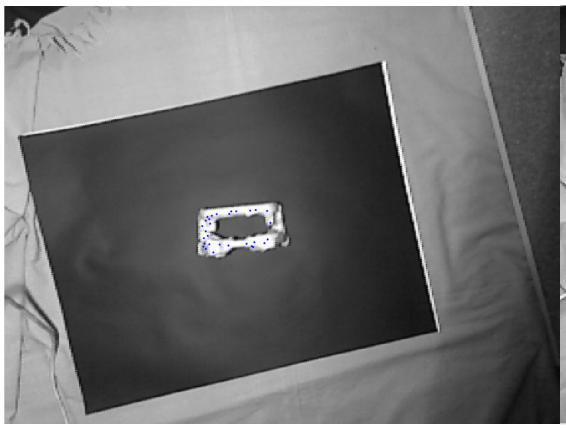


Figure 6.7: Mesh with distance method

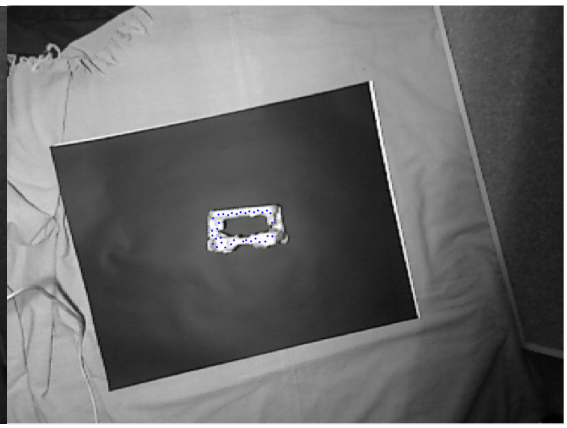


Figure 6.8: Mesh with the squares method

The object represented in Figures 6.7 and 6.8 represents the extreme of what this project was made to find. It has high gradients zones where it is almost impossible for the Kinect to have high resolution, so, as some of the point cloud are missing, and this effect affects the mesh on the gradient distance method. The division in squares method, since it uses both the objects location and the medium gradient, can surpass the dependency of a point cloud that the gradient distance method has, being able to represent a mesh in the entire object.

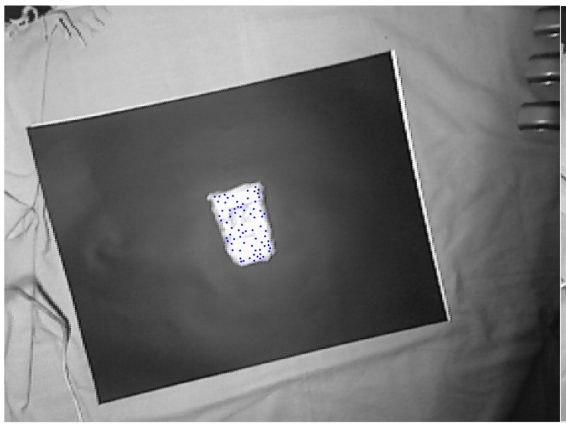


Figure 6.9: Mesh with distance method

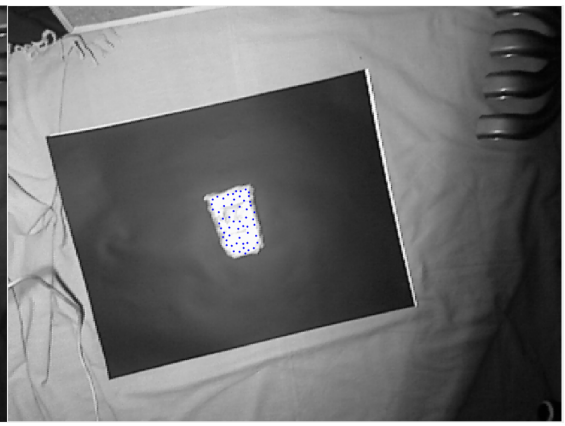


Figure 6.10: Mesh with the squares method

The third object, Figures 6.9 and 6.10 has an almost flat surface with some gradient on the edges, so the mesh looks regular on its surface. The object is a back cover from an electronic device.

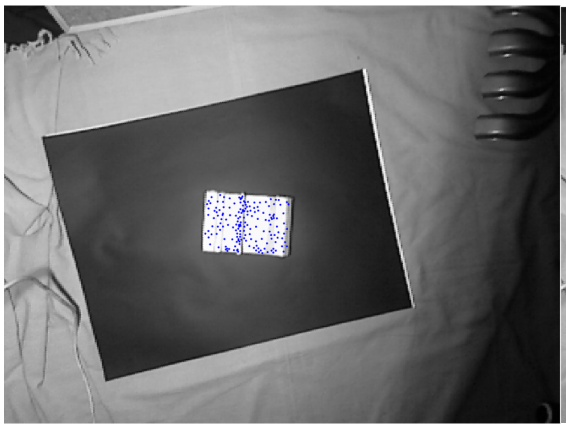


Figure 6.11: Mesh with distance method

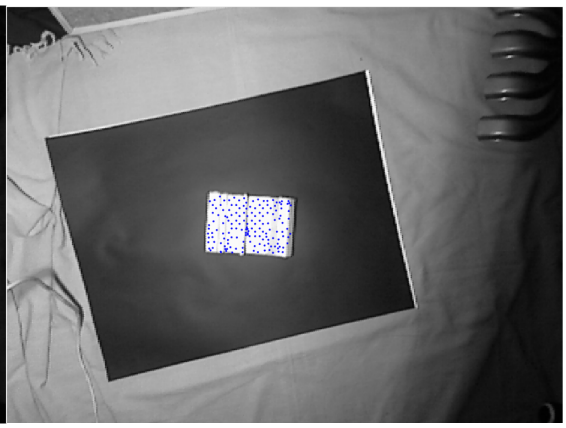


Figure 6.12: Mesh with the squares method

Figures 6.11 and 6.12 represent the results for an object with a flat surface with a high gradient zone in the middle, and this effect can be easily seen by the gradient distance method. The division in squares method, since it calculates the medium gradient, as most of the object were inside the defined interval, the mesh looks constant.

6.3 Components

6.3.1 Digital Potentiometers

As mentioned on Chapter 2, digital potentiometers were first used to convert the digital signal from the Arduino into an analog signal. The potentiometers used, only allow around 60 000 changes per second, and since the galvanometer can do 20 000 points per second the number of changes is superior to the number of points the galvanometer can perform. The disadvantage of the potentiometer is that these 60 thousand changes, are incremental, what means that in order to get from the biggest number, 255, to the smallest, 0, in one positive axis, 256 changes had to be made. Since 20 frames per second is the minimum required amount for a continuous effect on the eye, the 60 000 points drop to 3000. This means that with 7 measurement points, in a reading order from the top to the bottom and going back to the top would take the 3000 changes mentioned.

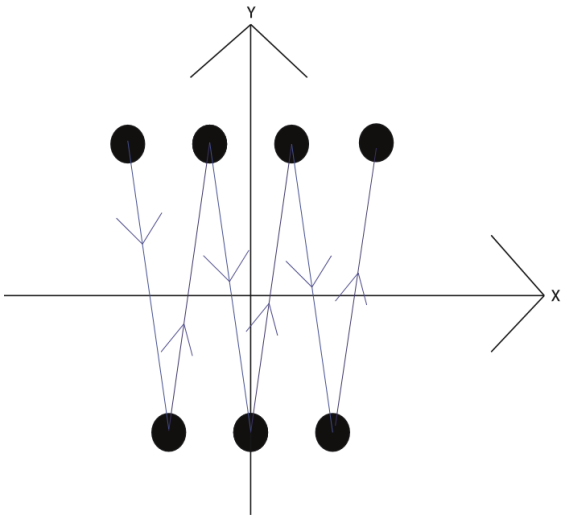


Figure 6.13: Example of a seven points movement by the galvanometer

At high speeds, doing max and min in just the positive axis, 0 to 255, as fast as the potentiometer could on a 32-kHz frequency, the wave form on the oscilloscope, sometimes had some fallings when it was going from 0 to 255 and vice versa, which means that instead of going just up, since the galvanometer reacts to fast changes on the signal, the mirror where the signal was being applied would stop, go down a little bit, stop and then go up again.

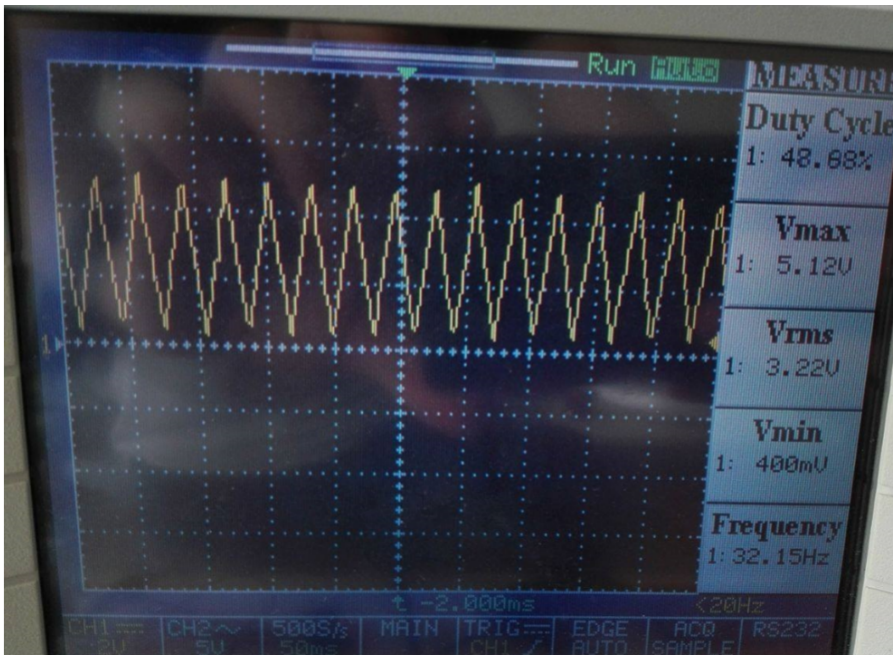


Figure 6.14: Up and down sequence at 32kHz

Another problem with the potentiometers was that they work in steps, which means

that if the intended point was in the middle of two steps, for example steps 1V and 2V and the intended point being 1.5V, it would be impossible to reach. This comparison is exaggerated and does not represent the actual steps as it is just an example. The actual voltage steps are represented on the attachments section. With the steps, it was noticed that they did not always had the same values and that the up-direction steps were different from the down-direction.

6.3.2 PWM, PWM Shield and DAC shield

The PWM shield has four capacitors and four resistances for the low pass filters, since there is a certain tolerance in each one, ten percent tolerance in the resistors, the result using each filter is a little different since the galvanometer can detect those changes. These changes can be visible by performing a square, using the limit's button on the GUI. It was observed that the square produced with the DAC shield is smother than with the PWM shield.

Without the PWM Shield, a PWM signal alone without the low pass filters could still do the work, but it would require a delay of 1 milisecond between each point.

Without this delay, the drawing would stop being points and would start being a random drawing as the galvanometer tries to read the PWM signal. Besides the 1 ms delay, it also requires a higher than 20kHz frequency as less than that would result in the representation of both limits of the digital signal, 0v and 5v.

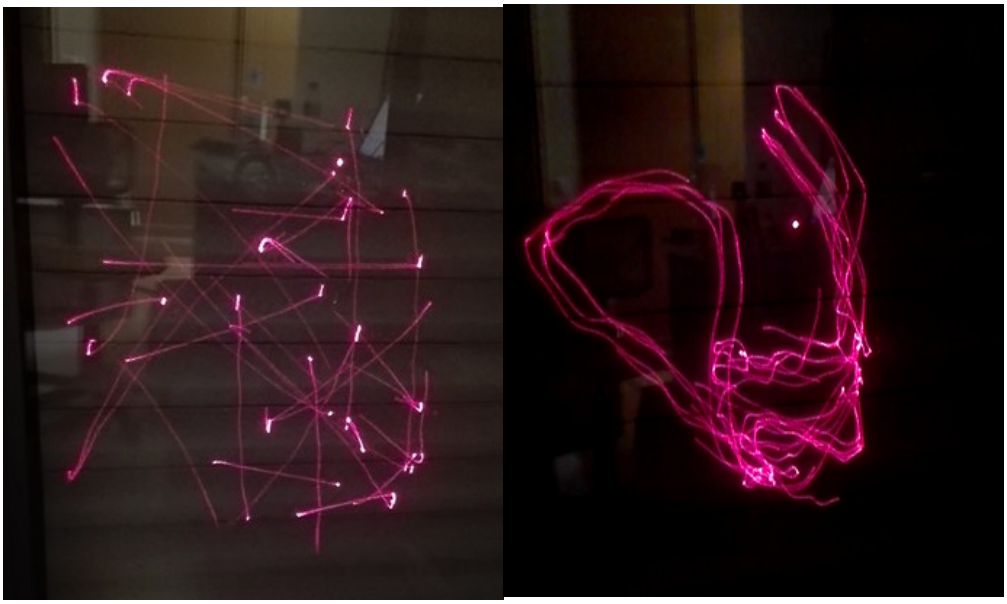


Figure 6.15: Mesh draw by the galvanometer

Figure 6.16: Drawing with a pause smaller than 1ms between points

Without a pause between the representation of the points, both PWM methods would represent in just a point as the galvanometer has no time to move. By measuring a for cycle that draws a 40 points square on the galvanometer, without pauses, the cycle time is 1812 microseconds, which does not mean that 80 points would take around 4000

microseconds since it is not linear, but it represents the need for pauses during the cycle, otherwise the galvanometer would not have time to represent the points, since 40 points in 1.812ms means 1 324 503 points per second on the galvanometer, which it can not draw.

To represent 20 000 points per second with 20 frames per second, it would represent 1000 points in 50ms, which means a 50 microsecond delay between points. Using this delay on the Arduino when trying to perform a 40 points square, the 50 microsecond delay was insufficient, since the galvanometer could not draw the edges of the square. Instead, 140 microseconds was the minimum amount of time to do it.

With a 140 microseconds delay, the laser would turn on before the galvanometer reach the intended measurement point. In order to calibrate the on time of the laser pointer with the coordinates, the necessary delay was 700 microseconds. This delay, with a 50 microseconds pause for the red pointer to make it visible, results in a drop from 20 000 points to 66 points. This pauses and maximum number of points were tuned for the PWM shield.

With the DAC shield, all the PWM characteristics were imported, and since the PWM shield has a close to analog signal, the speed characteristics are expected to be very similar.

Figure 6.17: GUI representation of the points

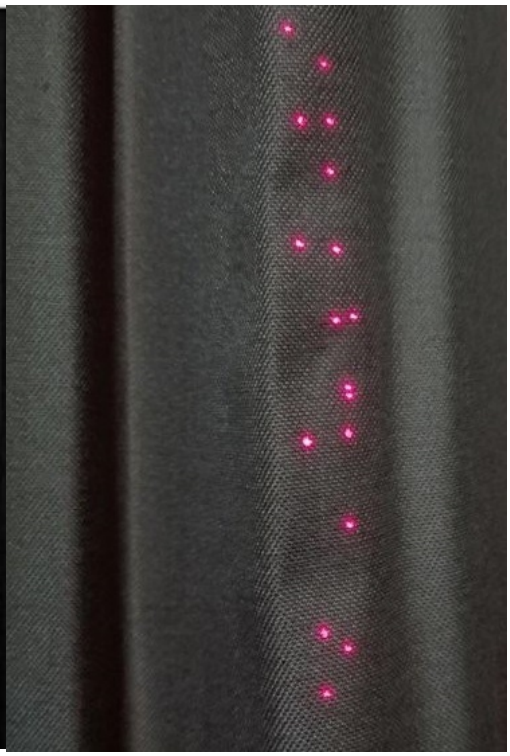


Figure 6.18: Points drawn by the galvanometer

Figure 6.19: GUI representation of the points

Figure 6.20: Points drawn by the galvanometer

The points drawn on the Figures 6.18 and 6.20 used the calibrations between the Kinect and the galvanometer but, some discrepancies occurred, due to a bad calibration of the galvanometer drivers.

Galvanometer drivers calibrations

The first guess for the discrepancy was an error in the calibration calculations but soon, it was observed that the square that was drawn using the GUI button for that effect, was not a square but a rectangle. The x and y axis did not had the same dimensions. To first measure the effect, the green laser was turned on, doing a line from the minimum -x to maximum +x. The edges of the line as well as the middle point were drawn so it was possible to measure the angle, which was indeed smaller than 20 degrees in both xy directions, as the biggest angle had 15 degrees. This angles invalidate the 20 degrees cross multiplication on the Arduino, being necessary a way to calibrate the laser to the desirable angle.

To calibrate the galvanometer a platform was built on which the representation of where the 20 degress should be was drawn.

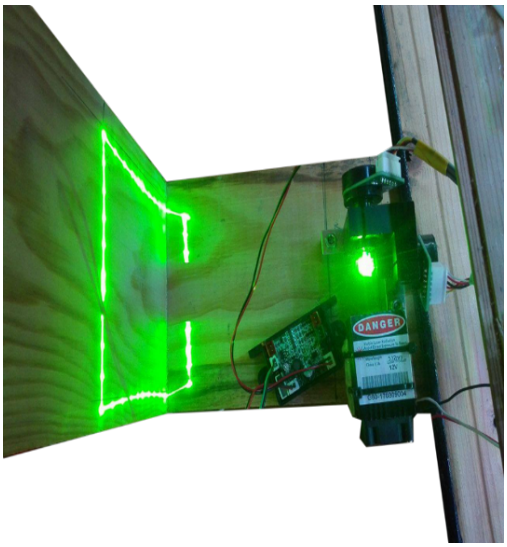


Figure 6.21: Top view of the calibrations base

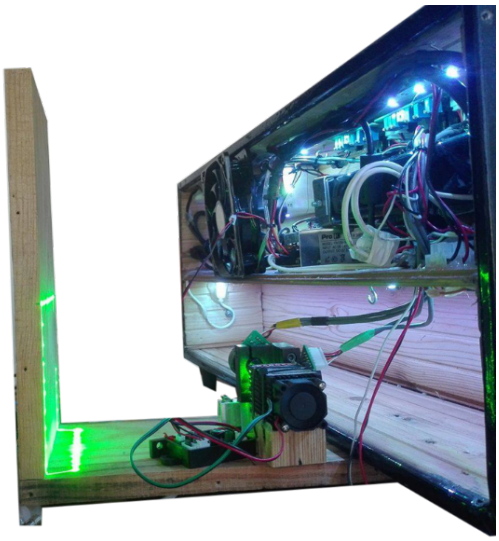


Figure 6.22: Side view of the calibrations base

The adjustment of the galvanometer is done by rotation the six potentiometer screws on each control drive. Since each of the potentiometer controls a different characteristic on the galvanometer, it was hard to calibrate, as a special pattern would need to be drawn which is the ILDA test patter.

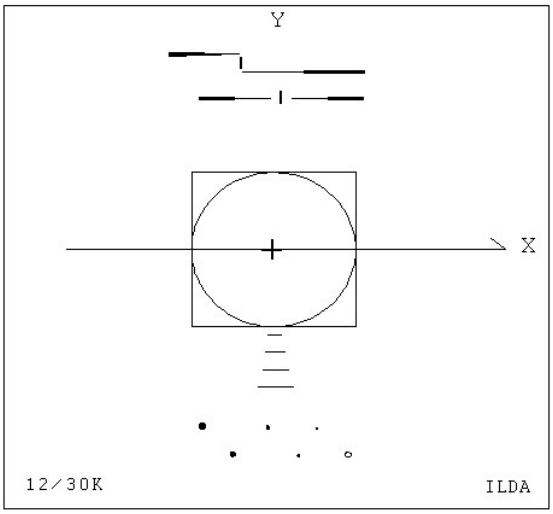


Figure 6.23: ILDA test pattern

Each line on the test pattern, tests a different characteristic, and in order to have a calibrated galvanometer, the test pattern needs to appear the same way as in the previous figure.

The test pattern coordinates can be downloaded from the laserfx website[105] as well as the on and off position of the laser. The final aspect of the laser calibrations

coordinates, with the on and off paths, on as green and off as blue, is represented in Figure 6.24.

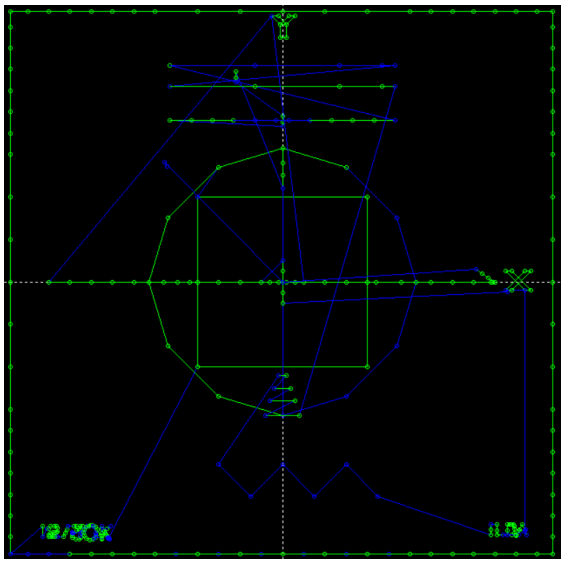


Figure 6.24: Drawing of the coordinates ILDA pattern

In order to represent those points, which are 1191 points, three arrays need to be done, laser, x and y axis, and then, they need to be read in a for cycle. Due to the high number of points, this task is impossible to perform with an Arduino Uno with the current configuration, as memory issues occur, so only the dimention of both axis was calibrated. The Figure 6.31 was done using the program mkv2[106], using the indented coordinates.

6.4 Calibration Results

The final results with the galvanometer axis calibrated can be seen on the next figures but due to other calibrations parameters being incorrect, some discrepancies occurred close to the limits of the galvanometer.

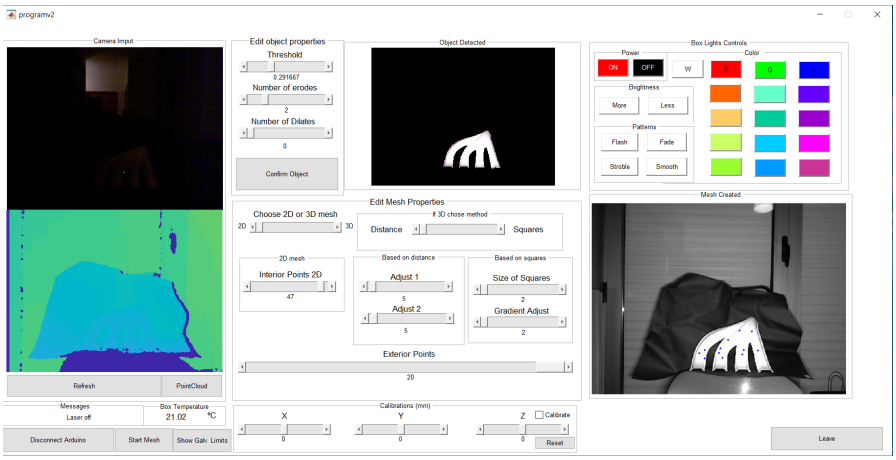


Figure 6.25: GUI fields

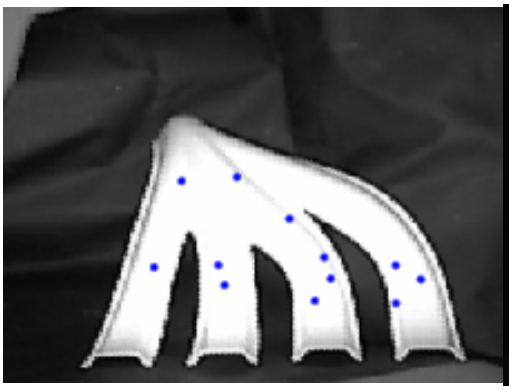


Figure 6.26: Points on the GUI

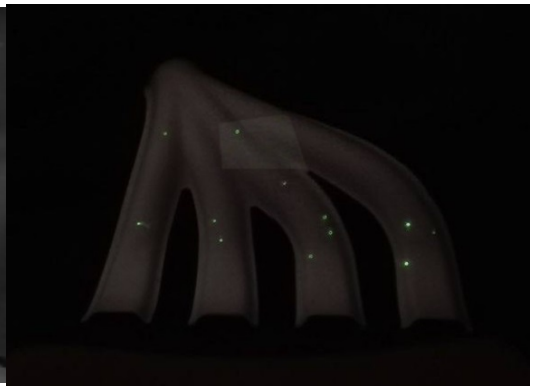


Figure 6.27: Points drawn by the laser

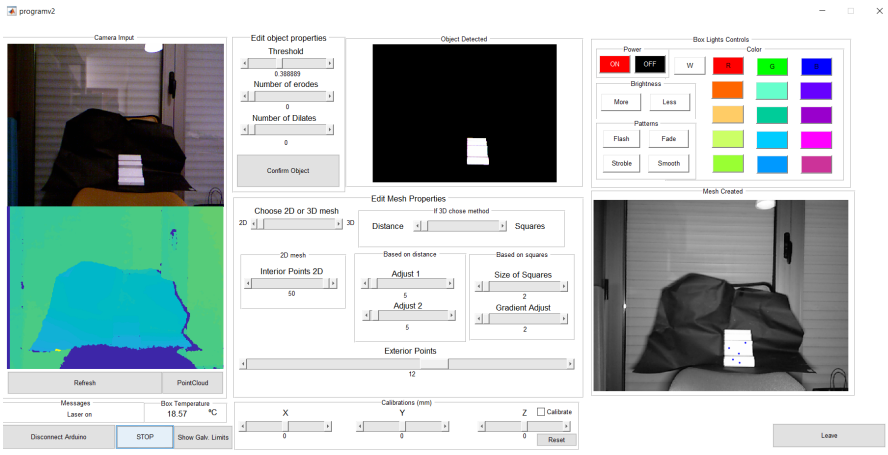


Figure 6.28: GUI fields

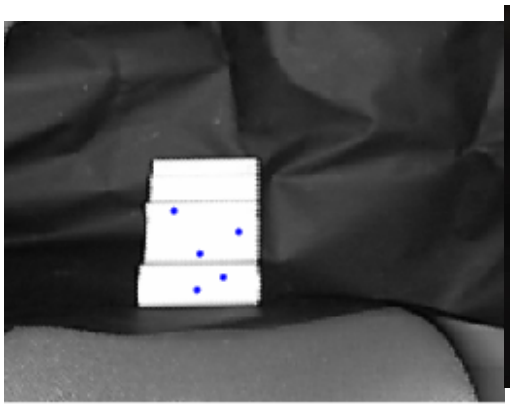


Figure 6.29: Points on the GUI

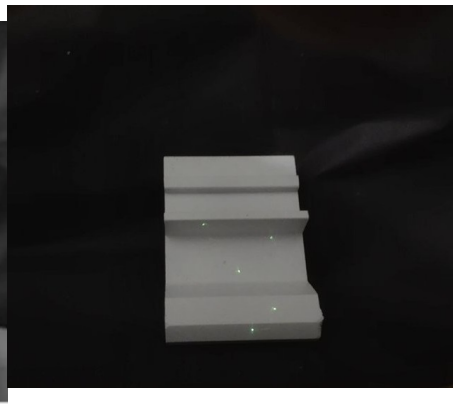


Figure 6.30: Points drawn by the laser

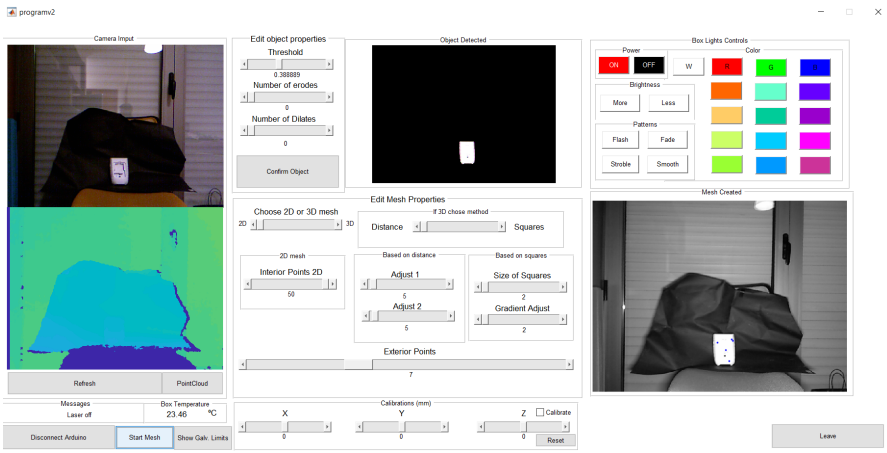


Figure 6.31: GUI fields

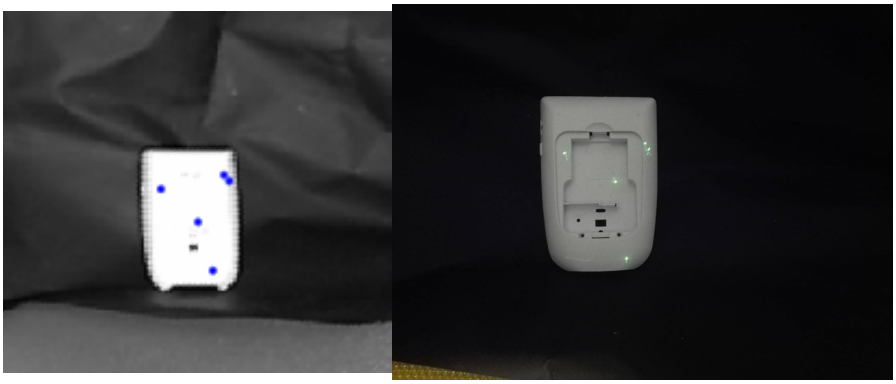


Figure 6.32: Points on the GUI

Figure 6.33: Points drawn by the laser

Chapter 7

Conclusions and Future Work

7.1 Conclusions

A LDV is a tool that can be used for multiple purposes, from vibration measurements on structures, to banking operations, and health care. The commercial solutions available are more focused in the measurement itself, but not in the choice of those points. The product created intends to solve the manual selection requirement of measurement points, increasing the mesh density in zones with high gradient, allowing an incorporation with the available vibrometer which is a single point vibrometer.

Since the available vibrometer is a single point vibrometer, a galvanometer used to deflect the laser beam and it is controlled by an Arduino, which communicates with a GUI developed in MATLAB.

The current product allows only a mesh of 21 points as the Arduino struggles to separate a bigger string, as each point is in the float format.

To control the galvanometer which uses analog signals, DACs and PWM methods were used, and one Arduino shield was built for each. With PWM, only the PWM signal was not enough as smaller than one millisecond delays between points would be read as noise. The problem was solved with low pass filters which were placed after a logic-gate to correct the signal coming from it. As for the DAC, four were necessary and only two addresses were possible, so, they were divided in pairs, two for each axis, and they are turned on and off when required.

No big difference between both shield was found, the only visible change was, when testing the galvanometer limits, the lines of the square were more regular with the DACs, which can be related to the tolerances on the low pass filter which can be 10 percent in the resistors.

Digital potentiometers were also tried but due to the fact that they work in steps and would require checking which was the closest number, made it not valid for this work.

Besides the steps, which changed with the direction, the speed of the step changes would result in only 7 points in the worst case, which is traveling between extremes.

To choose the points for measurement, a GUI in MATLAB was built which sends the signals to the Arduino via serial communication to control the galvanometer. In the GUI, the object, number of points, com port and interior lights color in the prototype can be regulated, it is also possible to change between depth cam and point cloud to check if the object is detected.

A prototype box was also built to fix all components and make the calibration values

between the Kinect and the galvanometer constant. To keep the temperature of the drivers constant, a fan was added to the prototype which keeps room temperature inside the prototype.

The potentiometers of the galvanometer drivers were also changed to calibrate the distances of both axis, making it 20 degrees for each axis, even though dampening and other calibrations still need to be fixed. With the distances fixed, calibrations between the Kinect and the galvanometer were done, which in tests, showed a close approach between the GUI points and the drawn points.

7.2 Future Work

The current work has a couple of limitations to which some solutions were thought to overcome them.

First, to have the galvanometer at its maximum performance, it is necessary to calibrate it so the ILDA test pattern will need to be drawn. As the Arduino is not capable of dealing with that number of points, it is necessary the use of other device as for example a NodeMCU which has 4Mb of memory.

To increase the current number of points that the Arduino can read, the message creation in MATLAB should send first all the x angles with a terminator and then the y angles, change the Arduino code to save two strings, one with the x angles and other with the y angles. This way only 4 divisions would be done, lights, limits, number of angles and then the angles. Other solution could be, instead of sending a mesh, send one point at a time, to perform the measurement, this way the limitations with the number of points disappear.

Automatic calibrations could also be introduced, as the current ones require manual measurements with a protractor and it is necessary to manually open a point cloud to check the distance from the zero of the galvanometer and the zero of the camera.

There is still the need to test the system with the vibrometer and create specific buttons on the GUI to trigger it as well as the Arduino code and tests with the vibrometer are also necessary to see if the vibration of the galvanometer mirrors, affects the final readings, and if so, if it can be removed.

References

[1] Heinz, S. (2011, February). Bringing Ultrasonic Fatigue to Light. Retrieved from http://www.polytec.com/fileadmin/user_uploads/Applications/Research_Engineering/Documents/OM_AN_InFocus_0211_Ultrasonic_Fatigue_E.pdf(visited on 12/06/2017)

[2] GmbH, P. (2008, November). Non-contact Vibration Measure- ments on Wind Power Plants. Retrieved from http://www.polytec.com/fileadmin/user_uploads/Applications/General_Vibrometry/OM_AN_VIB-G-10_Wind_Powerplant_2008_11_E.pdf(visited on 11/06/2017)

[3] GmbH, P. (2008, June). Measurement of Security Features on Banknotes. Retrieved from http://www.polytec.com/fileadmin/user_uploads/Solutions/Surface_Profiling/Documents/OM_AN_TOP-03_Banknotes_2008_06_E.pdf(visited on 11/06/2017)

[4] GmbH, P. (2017, June). Basic Principles of Vibrometry. Retrieved from http://www.polytec.com/us/solutions/vibration-measurement/basic-principles-of-vibrometry/(visited on 05/06/2017)

[5] Casacanditella, Luigi. (2016, March 07). NON-CONTACT CONTINUOUS RECORDING OF BLOOD PRESSURE PULSE WAVEFORM BY MEANS OF VIBROCARDIOGRAPHY. Retrieved from https://www.researchgate.net/publication/292629003_Non-Contact_Continuous_Recording_of_Blood_Pressure_Pulse_Waveform_By_Means_of_Vibrocardiography(visited on 09/06/2017)

[6] Luo, X. (2017, February 24). Galvanometer scanning technology for laser additive manufacturing Retrieved from https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10095/1/Galvanometer-scanning-technology-for-laser-additive-manufacturing/10.1117/12.2252973.short?SSO=1(visited on 10/05/2017)

[7] dwssystems. (2017, February 24). Additive Manufacturing Retrieved from http://dwssystems.com/wp-content/uploads/2014/02/galvanometro.jpg(visited on 10/08/2017)

[8] Yuan, Y. (2012, December 11). Optical-resolution photoacoustic microscopy for imaging blood vessels in vivo Retrieved from https://www.spiedigitallibrary.org/conference-proceedings-of-spie/8553/1/

Optical-resolution-photoacoustic-microscopy-for-imaging-blood-vessels-in-vivo/10.1117/12.999515.short(visited on 10/07/2017)

[9] Microsoft, Kinect for Windows Sensor Components and Specifications Retrieved from https://msdn.microsoft.com/en-us/library/jj131033.aspx(visited on 11/07/2017)

[10] Griffin, A. (2016, April 21). Xbox 360 production stopped as microsoft announces that console is going to be killed off Retrieved from https://msdn.microsoft.com/en-us/library/jj131033.aspx(visited on 12/07/2017)

[11] Zugara, Kinect 1 vs Kinect 2 Tech Comparison Retrieved from http://zugara.com/wp-content/uploads/Kinect-1-vs-Kinect-2-Tech-Comparison.png(visited on 13/07/2017)

[12] slidesharecdn, Kinect 1 vs Kinect 2 Tech Comparison Retrieved from https://image.slidesharecdn.com/kinectv2introductionandtutorial-141114042655-conversion-gate01/95/kinect-v2-introduction-and-tutorial-6-638.jpg?cb=1445117505(visited on 14/07/2017)

[13] arduino, Arduino Uno Rev3 Retrieved from https://store.arduino.cc/arduino-uno-rev3(visited on 14/06/2017)

[14] arduino, begin() Retrieved from https://www.arduino.cc/en/Serial/Begin(visited on 15/07/2017)

[15] Everlight, Technical Data Sheet 5mm Infrared LED Retrieved from https://cdn-shop.adafruit.com/datasheets/IR333_A_datasheet.pdf(visited on 01/06/2017)

[16] Microchip Technology Inc, MCP4725 Retrieved from https://cdn.sparkfun.com/datasheets/BreakoutBoards/MCP4725_2009.pdf(visited on 05/06/2017)

[17] How to Mechatronics, How I2C Communication Works and How To Use It with Arduino Retrieved from http://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/(visited on 05/06/2017)

[18] Hitech_components_llc, 5 PCs MC74HC00AN 74HC00AN (REPLACING FOR 74HC00N SN74HC00N HD74HC00P), Retrieved from http://www.ebay.ie/itm/5-PCs-MC74HC00AN-74HC00AN-REPLACING-FOR-74HC00N-SN74HC00N-HD74HC00P-/151841781737?hash=item235a79b7e9:g:SAoAAOSwvc9WFbav(visited on 05/06/2017)

[19] Texas Instruments,LM35 Precision Centigrade Temperature Sensors Retrieved from http://www.ti.com/lit/ds/symlink/lm35.pdf(visited on 10/05/2017)

[20] Cooler Master,SickleFlow 120 2000 RPM Blue LED Retrieved from http://www.coolermaster.com/cooling/case-fan/sickleflow-120-2000rpm-blue-led/(visited on 20/05/2017)

[21] Leroy Merlin,Kit tira LED Eglo 3 M Retrieved from http://www.leroymerlin.es/fp/16716966/kit-tira-led-eglo-3-m#ficha-tecnica(visited on 20/05/2017)

[22] prokelectronics,FAI12V-3A(C) Retrieved from http://www.prokelectronics.com/prok/index.php?option=com_virtuemart&view=productdetails&page=shop.product_details&flypage=flypage.tpl&category_id=366&product_id=702(visited on 20/05/2017)

[23] Laserwinkel, High speed Galvo System RGB-SCAN20 close-loop scanner Retrieved from: http://www.laserwinkel.nl/bestanden/20kpps.pdf (visited on 01/06/2017)

[24] Staples, Staples 500 Laser Pointer, Projects up to 500 yds Retrieved from: https://www.staples.com/Staples-500-Laser-Pointer-Projects-up-to-500-yds-17488-CC-/product_801990 (visited on 01/06/2017)

[77] Farnell, X9C103 datasheet Retrieved from: http://www.farnell.com/datasheets/31917.pdf (visited on 01/06/2017)

[26] Arduino Modules, KY-008 Laser Transmitter Module Retrieved from: http://arduinomodules.info/ky-008-laser-transmitter-module/ (visited on 06/06/2017)

[27] Lelong, KY-008 LASER TRANSMITTER SENSOR Retrieved from:https://www.lelong.com.my/ky-008-laser-transmitter-sensor-arduino-uno-compatible-nadieleczone-190676354-2018-03-htm (visited on 06/06/2017)

[28] Instructables, Keyes KY-008 Laser Transmitter Demystified Retrieved from: http://www.instructables.com/id/Keyes-KY-008-Laser-Transmitter-Demystified/ (visited on 06/06/2017)

[30] Ebay, Módulo Láser Verde G80 532nm 80mW para Escenario Luz/Lab/fuente de luz/12V/TTL/FAN Retrieved from: http://www.ebay.es/itm/G80-532nm-80mW-Green-Laser-Module-for-Stage-Light-Lab-Light-Source-12V-TTL-FAN-/231925756702?hash=item35ffda331e:g:SmwAAOSwKfVXIDy8 (visited on 08/08/2017)

[30] Wikipedia, Laser safety Retrieved from: https://en.wikipedia.org/wiki/Laser_safety (visited on 08/08/2017)

[31] Fritzing, Laser safety Retrieved from: http://fritzing.org/static/img/fritzing-preview-bb.png (visited on 08/08/2017)

[32] Fritzing, Fritzing electronics made easy Retrieved from: http://fritzing.org/home/ (visited on 08/08/2017)

[33] MATHWORKS, Image Acquisition Toolbox Support Package for Kinect For Windows Sensor Retrieved from: https://www.mathworks.com/matlabcentral/fileexchange/

40445-image-acquisition-toolbox-support-package-for-kinect-for-windows-sensor?requestedDomain=www.mathworks.com (visited on 08/08/2017)

[34] MATHWORKS, videoinput Retrieved from: https://www.mathworks.com/help/imaq/videoinput.html (visited on 01/06/2017)

[35] MATHWORKS, getsnapshot Retrieved from: https://www.mathworks.com/help/imaq/getsnapshot.html (visited on 08/08/2017)

[36] A, Anwer.(2014, September 12), Kinect Depth Normalization Retrieved from: https://www.mathworks.com/matlabcentral/fileexchange/47830-kinect-depth-normalization (visited on 08/08/2017)

[37] MATHWORKS, im2bw, Retrieved from: https://www.mathworks.com/help/images/ref/im2bw.html (visited on 08/08/2017)

[38] MATHWORKS, rgb2gray, Retrieved from: https://www.mathworks.com/help/matlab/ref/rgb2gray.html (visited on 08/08/2017)

[39] MATHWORKS, pcfromkinect, Retrieved from: https://www.mathworks.com/help/vision/ref/pcfromkinect.html (visited on 08/08/2017)

[40] MATHWORKS, polyxpoly, Retrieved from: https://www.mathworks.com/help/map/ref/polyxpoly.html (visited on 08/08/2017)

[41] MATHWORKS, gradient, Retrieved from: https://www.mathworks.com/help/matlab/ref/gradient.html (visited on 08/08/2017)

[42] MATHWORKS, bwboundaries, Retrieved from: https://www.mathworks.com/help/images/ref/bwboundaries.html (visited on 08/08/2017)

[43] MATHWORKS, edge, Retrieved from: https://www.mathworks.com/help/images/ref/edge.html (visited on 01/06/2017)

[44] MATHWORKS, delaunay, Retrieved from: https://www.mathworks.com/help/matlab/ref/delaunay.html (visited on 01/06/2017)

[45] Vincent, (2008, September 1), Cartesian coordinates and transformation matrices, Retrieved from: http://polymathprogrammer.com/2008/09/01/cartesian-coordinates-and-transformation-matrices/ (visited on 01/06/2017)

[46] MATWORKS, atand, Retrieved from: https://www.mathworks.com/help/matlab/ref/atand.html (visited on 01/06/2017)

[47] Arduino, begin(), Retrieved from: https://www.arduino.cc/en/Serial/Begin (visited on 01/06/2017)

[48] MATHWORKS, fopen, Retrieved from: https://www.mathworks.com/help/matlab/ref/fopen.html (visited on 01/06/2017)

[49] MATHWORKS, fprintf, Retrieved from: https://www.mathworks.com/help/matlab/ref/fprintf.html (visited on 01/06/2017)

[50] MATHWORKS, fgets, Retrieved from: https://www.mathworks.com/help/matlab/ref/fgets.html (visited on 01/06/2017)

[51] MATHWORKS, fclose, Retrieved from: https://www.mathworks.com/help/matlab/ref/fclose.html (visited on 01/06/2017)

[52] MATHWORKS, try, Retrieved from: https://www.mathworks.com/help/matlab/ref/try.html (visited on 01/06/2017)

[53] Arduino, setup, Retrieved from: https://www.arduino.cc/en/Reference/Setup (visited on 01/06/2017)

[54] Arduino, loop, Retrieved from: https://www.arduino.cc/en/Reference/Loop (visited on 01/06/2017)

[55] Arduino, SerialEvent, Retrieved from: https://www.arduino.cc/en/Tutorial/SerialEvent (visited on 01/06/2017)

[56] Arduino, Begin, Retrieved from: https://www.arduino.cc/en/Serial/Begin (visited on 01/06/2017)

[57] Arduino, Read, Retrieved from: https://www.arduino.cc/en/Serial/Read (visited on 01/06/2017)

[58] Arduino, Print, Retrieved from: https://www.arduino.cc/en/Serial/Print (visited on 01/06/2017)

[59] MATHWORKS, Events and Callbacks, Retrieved from: https://www.mathworks.com/help/matlab/matlab_external/events-and-callbacks.html (visited on 01/06/2017)

[60] Arduino, ReadStringUntil, Retrieved from: https://www.arduino.cc/en/Serial/ReadStringUntil (visited on 01/06/2017)

[61] Stackoverflow, Split String into String array, Retrieved from: https://stackoverflow.com/questions/9072320/split-string-into-string-array (visited on 01/06/2017)

[62] Arduino, analogWrite(), Retrieved from: https://www.arduino.cc/en/Reference/AnalogWrite (visited on 01/06/2017)

[63] Arduino, PWM, Retrieved from: https://www.arduino.cc/en/Tutorial/PWM (visited on 01/06/2017)

[64] Arduino, PwmFrequency, Retrieved from: https://playground.arduino.cc/Code/PwmFrequency (visited on 01/06/2017)

[65] Arduino, PWM frequency library, Retrieved from: https://forum.arduino.cc/index.php?topic=117425.0 (visited on 01/06/2017)

[66] Sparkfun, I2C, Retrieved from: https://learn.sparkfun.com/tutorials/i2c (visited on 01/06/2017)

[67] Adafruit, Using with Arduino, Retrieved from: https://learn.adafruit.com/mcp4725-12-bit-dac-tutorial/using-with-arduino (visited on 01/06/2017)

[68] Arduino, Wire, Retrieved from: https://www.arduino.cc/en/Reference/Wire (visited on 01/06/2017)

[69] Sparkfun, MCP4725 Digital to Analog Converter Hookup Guide, Retrieved from: https://learn.sparkfun.com/tutorials/mcp4725-digital-to-analog-converter-hookup-guide (visited on 01/06/2017)

[70] Wikipedia, Vibration. [Online]. Available: https://en.wikipedia.org/wiki/Vibration (visited on 01/06/2017)

[71] IMV,What is the need for vibration measurement [Online]. Available: https://www.imv.co.jp/e/pr/story/main04_1.php (visited on 01/06/2017)

[72] Hindawi,Shock and Vibrations [Online]. Available: https://www.hindawi.com/journals/sv/2002/968509/abs/ (visited on 01/06/2017)

[73] OMS,OMS-Products-Multibeam Vibrometer [Online]. Available: http://www.omscorporation.com/products/multibeam-vibrometer/ (visited on 01/06/2017)

[74] polytec,Single-Point Vibromters [Online]. Available: http://www.polytec.com/us/products/vibration-sensors/single-point-vibrometers/ (visited on 01/06/2017)

[75] HowStuffWorks, The Kinect Sensor [Online]. Available: http://electronics.howstuffworks.com/microsoft-kinect2.htm (visited on 01/06/2017)

[76] Microsoft Developer Network, Kinect Sensor [Online]. Available: https://msdn.microsoft.com/en-us/library/hh438998.aspx (visited on 01/06/2017)

[77] Farnell, X9C103 datasheet [Online]. Available: http://www.farnell.com/datasheets/31917.pdf (visited on 01/06/2017)

[78] Youtube, Digital Potentiometers - Short Circuits Episode3 [Online]. Available: https://www.youtube.com/watch?v=iOINcW8BP3w (visited on 01/06/2017)

[79] Arduino,Software [Online]. Available: https://www.arduino.cc/en/Main/Software (visited on 01/06/2017)

[80] Mathworks,matlab-gui [Online]. Available: https://www.mathworks.com/discovery/matlab-gui.html (visited on 01/06/2017)

[81] damow,Building a Laser Show [Online]. Available: https://damow.net/building-a-laser-show/ (visited on 01/06/2017)

[82] Wikipedia,Mirror galvanometer [Online]. Available: https://en.wikipedia.org/wiki/Mirror_galvanometer (visited on 01/06/2017)

[83] SKANECT,3D Scanning [Online]. Available: http://skanect.occipital.com/ (visited on 01/06/2017)

[84] SKANECT,Features[Online]. Available: http://skanect.occipital.com/features/ (visited on 01/06/2017)

[85] MetroLaser,VibroMet MB-LDV[Online]. Available: http://www.sv-china.com/uploadmyfile/20100312123058.pdf (visited on 01/06/2017)

[86] polytec,RoboVid[Online]. Available: http://www.polytec.com/fileadmin/user_uploads/Applications/Automotive_Transportation/Documents/OM_AN_InFocus_0108_RoboVib_E.pdf (visited on 01/06/2017)

[87] Youtube,PSV Laser Scanning Vibrometer[Online]. Available: https://www.youtube.com/watch?v=M3HC_G59t7o (visited on 01/06/2017)

[88] NetZeroTools,Kanomax 4200 Compact Vibration Meter Accelerometer Sensor Monitor[Online]. Available: http://www.netzerotools.com/kanomax-4200-compact-easy-to-use-vibration-meter (visited on 01/06/2017)

[89] zugara, How Does The Kinect 2 Compare To The Kinect 1?[Online]. Available: http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1 (visited on 01/06/2017)

[90] arduino, arduinoBoardUno[Online]. Available: https://www.arduino.cc/en/main/arduinoBoardUno (visited on 01/06/2017)

[91] Byteway, Kinect overview before starting programming [Online]. Available: https://byteway.wordpress.com/tag/kinect-limitations/ (visited on 01/06/2017)

[92] Griffin, A. (2016, April 21) Xbox 360 production stopped as Microsoft announces that console is going to be killed off [Online]. Available: http://www.independent.co.uk/life-style/gadgets-and-tech/news/xbox-360-production-stopped-as-microsoft-announces-that-console-is-going-to-be-killed-html (visited on 01/06/2017)

[93] mathworks,arduino-matlab [Online]. Available: https://www.mathworks.com/hardware-support/arduino-matlab.html (visited on 01/06/2017)

[94] mathworks,outputbuffersize[Online]. Available: https://www.mathworks.com/help/instrument/outputbuffersize.html (visited on 01/06/2017)

[95] MATHWORKS, bwboundaries, Retrieved from: https://www.mathworks.com/help/images/ref/bwboundaries.html (visited on 01/06/2017)

[96] ArduinoInfo, Arduino-PWM-Frequency, Retrieved from: https://arduino-info.wikispaces.com/Arduino-PWM-Frequency (visited on 01/06/2017)

[97] Google code Archive, arduino-pwm-frequency-library, Retrieved from: https://code.google.com/archive/p/arduino-pwm-frequency-library/ (visited on 01/06/2017)

[98] TrewRoad, Instructables, X9C103P Basic Operation, Retrieved from: http://www.instructables.com/id/X9C103P-Basic-Operation/ (visited on 01/06/2017)

[99] MATHWORKS, Automatically Refresh Plot in a GUIDE App, Retrieved from: https://www.mathworks.com/help/matlab/creating_guis/automatically-refresh-plot-in-a-guide-gui.html (visited on 01/06/2017)

[100] MATHWORKS, Terminator, Retrieved from: https://www.mathworks.com/help/instrument/terminator.html (visited on 01/06/2017)

[101] MATHWORKS, Serial, Retrieved from: https://www.mathworks.com/help/matlab/ref/serial.html (visited on 01/06/2017)

[102] Guimarães, F.(2016, May 26) Usando Infravermelho para controlar um aparelho de TV, Retrieved from: http://arduinoccbr.blogspot.pt/2016/05/usando-infravermelho-para-controlar-um.html (visited on 01/06/2017)

[103] Monk,S. Timer Library for Arduino, Retrieved from: https://playground.arduino.cc/Code/Timer (visited on 01/06/2017)

[104] Pinterest, Pikachu pin, Retrieved from: https://br.pinterest.com/pin/677158493941264487/ (visited on 01/06/2017)

[105] laserfx, Laser Show Systems - Scanning Systems, Retrieved from: http://www.laserfx.com/Backstage.LaserFX.com/Systems/Scanning1.html (visited on 01/06/2017)

[106] Home Built Laser Projector, Frame Editor for Win32, Retrieved from: http://elm-chan.org/works/vlp/report_e.html (visited on 01/06/2017)

Appendices

Appendix A

Digital potentiometer Table

The following tables show the tap position, the analog value on the arduino analog pin, the correspondent voltage to that analog value, the correspondent resistance, and the actual resistance. [98]

Table A.1: Digital potentiometer values on descent 19 to 0

WiperDirection DOWN				
Tap = 19	ADC = 191	Voltage = 0.934	Tap Ohm = 1919	Law = 1867
Tap = 18	ADC = 181	Voltage = 0.885	Tap Ohm = 1818	Law = 1769
Tap = 17	ADC = 174	Voltage = 0.850	Tap Ohm = 1717	Law = 1701
Tap = 16	ADC = 162	Voltage = 0.792	Tap Ohm = 1616	Law = 1584
Tap = 15	ADC = 153	Voltage = 0.748	Tap Ohm = 1515	Law = 1496
Tap = 14	ADC = 145	Voltage = 0.709	Tap Ohm = 1414	Law = 1417
Tap = 13	ADC = 132	Voltage = 0.645	Tap Ohm = 1313	Law = 1290
Tap = 12	ADC = 121	Voltage = 0.591	Tap Ohm = 1212	Law = 1183
Tap = 11	ADC = 112	Voltage = 0.547	Tap Ohm = 1111	Law = 1095
Tap = 10	ADC = 102	Voltage = 0.499	Tap Ohm = 1010	Law = 997
Tap = 09	ADC = 91	Voltage = 0.445	Tap Ohm = 909	Law = 890
Tap = 08	ADC = 81	Voltage = 0.396	Tap Ohm = 808	Law = 792
Tap = 07	ADC = 72	Voltage = 0.352	Tap Ohm = 707	Law = 704
Tap = 06	ADC = 61	Voltage = 0.298	Tap Ohm = 606	Law = 596
Tap = 05	ADC = 49	Voltage = 0.239	Tap Ohm = 505	Law = 479
Tap = 04	ADC = 40	Voltage = 0.196	Tap Ohm = 404	Law = 391
Tap = 03	ADC = 29	Voltage = 0.142	Tap Ohm = 303	Law = 283
Tap = 02	ADC = 19	Voltage = 0.093	Tap Ohm = 202	Law = 186
Tap = 01	ADC = 9	Voltage = 0.044	Tap Ohm = 101	Law = 88
Tap = 00	ADC = 0	Voltage = 0.000	Tap Ohm = 0	Law = 0

Table A.2: Digital potentiometer values on descent 59 to 20

WiperDirection DOWN				
Tap = 59	ADC = 438	Voltage = 2.141	Tap Ohm = 5959	Law = 4282
Tap = 58	ADC = 438	Voltage = 2.141	Tap Ohm = 5858	Law = 4282
Tap = 57	ADC = 430	Voltage = 2.102	Tap Ohm = 5757	Law = 4203
Tap = 56	ADC = 430	Voltage = 2.102	Tap Ohm = 5656	Law = 4203
Tap = 55	ADC = 422	Voltage = 2.063	Tap Ohm = 5555	Law = 4125
Tap = 54	ADC = 421	Voltage = 2.058	Tap Ohm = 5454	Law = 4115
Tap = 53	ADC = 417	Voltage = 2.038	Tap Ohm = 5353	Law = 4076
Tap = 52	ADC = 414	Voltage = 2.023	Tap Ohm = 5252	Law = 4047
Tap = 51	ADC = 409	Voltage = 1.999	Tap Ohm = 5151	Law = 3998
Tap = 50	ADC = 406	Voltage = 1.984	Tap Ohm = 5050	Law = 3969
Tap = 49	ADC = 402	Voltage = 1.965	Tap Ohm = 4949	Law = 3930
Tap = 48	ADC = 401	Voltage = 1.960	Tap Ohm = 4848	Law = 3920
Tap = 47	ADC = 395	Voltage = 1.931	Tap Ohm = 4747	Law = 3861
Tap = 46	ADC = 391	Voltage = 1.911	Tap Ohm = 4646	Law = 3822
Tap = 45	ADC = 388	Voltage = 1.896	Tap Ohm = 4545	Law = 3793
Tap = 44	ADC = 386	Voltage = 1.887	Tap Ohm = 4444	Law = 3773
Tap = 43	ADC = 380	Voltage = 1.857	Tap Ohm = 4343	Law = 3715
Tap = 42	ADC = 377	Voltage = 1.843	Tap Ohm = 4242	Law = 3685
Tap = 41	ADC = 371	Voltage = 1.813	Tap Ohm = 4141	Law = 3627
Tap = 40	ADC = 367	Voltage = 1.794	Tap Ohm = 4040	Law = 3587
Tap = 39	ADC = 365	Voltage = 1.784	Tap Ohm = 3939	Law = 3568
Tap = 38	ADC = 359	Voltage = 1.755	Tap Ohm = 3838	Law = 3509
Tap = 37	ADC = 357	Voltage = 1.745	Tap Ohm = 3737	Law = 3490
Tap = 36	ADC = 349	Voltage = 1.706	Tap Ohm = 3636	Law = 3412
Tap = 35	ADC = 346	Voltage = 1.691	Tap Ohm = 3535	Law = 3382
Tap = 34	ADC = 339	Voltage = 1.657	Tap Ohm = 3434	Law = 3314
Tap = 33	ADC = 333	Voltage = 1.628	Tap Ohm = 3333	Law = 3255
Tap = 32	ADC = 320	Voltage = 1.564	Tap Ohm = 3232	Law = 3128
Tap = 31	ADC = 314	Voltage = 1.535	Tap Ohm = 3131	Law = 3069
Tap = 30	ADC = 305	Voltage = 1.491	Tap Ohm = 3030	Law = 2981
Tap = 29	ADC = 295	Voltage = 1.442	Tap Ohm = 2929	Law = 2884
Tap = 28	ADC = 283	Voltage = 1.383	Tap Ohm = 2828	Law = 2766
Tap = 27	ADC = 272	Voltage = 1.329	Tap Ohm = 2727	Law = 2659
Tap = 26	ADC = 265	Voltage = 1.295	Tap Ohm = 2626	Law = 2590
Tap = 25	ADC = 253	Voltage = 1.237	Tap Ohm = 2525	Law = 2473
Tap = 24	ADC = 243	Voltage = 1.188	Tap Ohm = 2424	Law = 2375
Tap = 23	ADC = 238	Voltage = 1.163	Tap Ohm = 2323	Law = 2326
Tap = 22	ADC = 222	Voltage = 1.085	Tap Ohm = 2222	Law = 2170
Tap = 21	ADC = 212	Voltage = 1.036	Tap Ohm = 2121	Law = 2072
Tap = 20	ADC = 202	Voltage = 0.987	Tap Ohm = 2020	Law = 1975

Table A.3: Digital potentiometer values on descent 99 to 60

WiperDirection DOWN				
Tap = 99	ADC = 994	Voltage = 4.858	Tap Ohm = 10000	Law = 9717
Tap = 98	ADC = 937	Voltage = 4.580	Tap Ohm = 9898	Law = 9159
Tap = 97	ADC = 890	Voltage = 4.350	Tap Ohm = 9797	Law = 8700
Tap = 96	ADC = 849	Voltage = 4.150	Tap Ohm = 9696	Law = 8299
Tap = 95	ADC = 814	Voltage = 3.978	Tap Ohm = 9595	Law = 7957
Tap = 94	ADC = 784	Voltage = 3.832	Tap Ohm = 9494	Law = 7664
Tap = 93	ADC = 757	Voltage = 3.700	Tap Ohm = 9393	Law = 7400
Tap = 92	ADC = 732	Voltage = 3.578	Tap Ohm = 9292	Law = 7155
Tap = 91	ADC = 711	Voltage = 3.475	Tap Ohm = 9191	Law = 6950
Tap = 90	ADC = 691	Voltage = 3.377	Tap Ohm = 9090	Law = 6755
Tap = 89	ADC = 672	Voltage = 3.284	Tap Ohm = 8989	Law = 6569
Tap = 88	ADC = 658	Voltage = 3.216	Tap Ohm = 8888	Law = 6432
Tap = 87	ADC = 640	Voltage = 3.128	Tap Ohm = 8787	Law = 6256
Tap = 86	ADC = 628	Voltage = 3.069	Tap Ohm = 8686	Law = 6139
Tap = 85	ADC = 614	Voltage = 3.001	Tap Ohm = 8585	Law = 6002
Tap = 84	ADC = 602	Voltage = 2.942	Tap Ohm = 8484	Law = 5885
Tap = 83	ADC = 591	Voltage = 2.889	Tap Ohm = 8383	Law = 5777
Tap = 82	ADC = 582	Voltage = 2.845	Tap Ohm = 8282	Law = 5689
Tap = 81	ADC = 571	Voltage = 2.791	Tap Ohm = 8181	Law = 5582
Tap = 80	ADC = 561	Voltage = 2.742	Tap Ohm = 8080	Law = 5484
Tap = 79	ADC = 553	Voltage = 2.703	Tap Ohm = 7979	Law = 5406
Tap = 78	ADC = 543	Voltage = 2.654	Tap Ohm = 7878	Law = 5308
Tap = 77	ADC = 535	Voltage = 2.615	Tap Ohm = 7777	Law = 5230
Tap = 76	ADC = 529	Voltage = 2.586	Tap Ohm = 7676	Law = 5171
Tap = 75	ADC = 522	Voltage = 2.551	Tap Ohm = 7575	Law = 5103
Tap = 74	ADC = 516	Voltage = 2.522	Tap Ohm = 7474	Law = 5044
Tap = 73	ADC = 510	Voltage = 2.493	Tap Ohm = 7373	Law = 4985
Tap = 72	ADC = 503	Voltage = 2.458	Tap Ohm = 7272	Law = 4917
Tap = 71	ADC = 499	Voltage = 2.439	Tap Ohm = 7171	Law = 4878
Tap = 70	ADC = 492	Voltage = 2.405	Tap Ohm = 7070	Law = 4809
Tap = 69	ADC = 488	Voltage = 2.385	Tap Ohm = 6969	Law = 4770
Tap = 68	ADC = 482	Voltage = 2.356	Tap Ohm = 6868	Law = 4712
Tap = 67	ADC = 476	Voltage = 2.326	Tap Ohm = 6767	Law = 4653
Tap = 66	ADC = 470	Voltage = 2.297	Tap Ohm = 6666	Law = 4594
Tap = 65	ADC = 467	Voltage = 2.283	Tap Ohm = 6565	Law = 4565
Tap = 64	ADC = 463	Voltage = 2.263	Tap Ohm = 6464	Law = 4526
Tap = 63	ADC = 457	Voltage = 2.234	Tap Ohm = 6363	Law = 4467
Tap = 62	ADC = 454	Voltage = 2.219	Tap Ohm = 6262	Law = 4438
Tap = 61	ADC = 450	Voltage = 2.199	Tap Ohm = 6161	Law = 4399
Tap = 60	ADC = 444	Voltage = 2.170	Tap Ohm = 6060	Law = 4340

Table A.4: Digital potentiometer values on ascent 0 to 19

WiperDirection UP				
Tap = 00	ADC = 0	Voltage = 0.000	Tap Ohm = 0	Law = 0
Tap = 01	ADC = 7	Voltage = 0.034	Tap Ohm = 101	Law = 68
Tap = 02	ADC = 17	Voltage = 0.083	Tap Ohm = 202	Law = 166
Tap = 03	ADC = 28	Voltage = 0.137	Tap Ohm = 303	Law = 274
Tap = 04	ADC = 38	Voltage = 0.186	Tap Ohm = 404	Law = 371
Tap = 05	ADC = 49	Voltage = 0.239	Tap Ohm = 505	Law = 479
Tap = 06	ADC = 60	Voltage = 0.293	Tap Ohm = 606	Law = 587
Tap = 07	ADC = 72	Voltage = 0.352	Tap Ohm = 707	Law = 704
Tap = 08	ADC = 79	Voltage = 0.386	Tap Ohm = 808	Law = 772
Tap = 09	ADC = 90	Voltage = 0.440	Tap Ohm = 909	Law = 880
Tap = 10	ADC = 101	Voltage = 0.494	Tap Ohm = 1010	Law = 987
Tap = 11	ADC = 110	Voltage = 0.538	Tap Ohm = 1111	Law = 1075
Tap = 12	ADC = 122	Voltage = 0.596	Tap Ohm = 1212	Law = 1193
Tap = 13	ADC = 132	Voltage = 0.645	Tap Ohm = 1313	Law = 1290
Tap = 14	ADC = 142	Voltage = 0.694	Tap Ohm = 1414	Law = 1388
Tap = 15	ADC = 152	Voltage = 0.743	Tap Ohm = 1515	Law = 1486
Tap = 16	ADC = 162	Voltage = 0.792	Tap Ohm = 1616	Law = 1584
Tap = 17	ADC = 171	Voltage = 0.836	Tap Ohm = 1717	Law = 1672
Tap = 18	ADC = 181	Voltage = 0.885	Tap Ohm = 1818	Law = 1769
Tap = 19	ADC = 193	Voltage = 0.943	Tap Ohm = 1919	Law = 1887

Table A.5: Digital potentiometer values on ascent 20 to 59

WiperDirection UP				
Tap = 20	ADC = 202	Voltage = 0.987	Tap Ohm = 2020	Law = 1975
Tap = 21	ADC = 216	Voltage = 1.056	Tap Ohm = 2121	Law = 2111
Tap = 22	ADC = 224	Voltage = 1.095	Tap Ohm = 2222	Law = 2190
Tap = 23	ADC = 235	Voltage = 1.149	Tap Ohm = 2323	Law = 2297
Tap = 24	ADC = 240	Voltage = 1.173	Tap Ohm = 2424	Law = 2346
Tap = 25	ADC = 254	Voltage = 1.241	Tap Ohm = 2525	Law = 2483
Tap = 26	ADC = 262	Voltage = 1.281	Tap Ohm = 2626	Law = 2561
Tap = 27	ADC = 273	Voltage = 1.334	Tap Ohm = 2727	Law = 2669
Tap = 28	ADC = 284	Voltage = 1.388	Tap Ohm = 2828	Law = 2776
Tap = 29	ADC = 294	Voltage = 1.437	Tap Ohm = 2929	Law = 2874
Tap = 30	ADC = 299	Voltage = 1.461	Tap Ohm = 3030	Law = 2923
Tap = 31	ADC = 313	Voltage = 1.530	Tap Ohm = 3131	Law = 3060
Tap = 32	ADC = 322	Voltage = 1.574	Tap Ohm = 3232	Law = 3148
Tap = 33	ADC = 332	Voltage = 1.623	Tap Ohm = 3333	Law = 3245
Tap = 34	ADC = 338	Voltage = 1.652	Tap Ohm = 3434	Law = 3304
Tap = 35	ADC = 348	Voltage = 1.701	Tap Ohm = 3535	Law = 3402
Tap = 36	ADC = 350	Voltage = 1.711	Tap Ohm = 3636	Law = 3421
Tap = 37	ADC = 356	Voltage = 1.740	Tap Ohm = 3737	Law = 3480
Tap = 38	ADC = 360	Voltage = 1.760	Tap Ohm = 3838	Law = 3519
Tap = 39	ADC = 367	Voltage = 1.794	Tap Ohm = 3939	Law = 3587
Tap = 40	ADC = 369	Voltage = 1.804	Tap Ohm = 4040	Law = 3607
Tap = 41	ADC = 372	Voltage = 1.818	Tap Ohm = 4141	Law = 3636
Tap = 42	ADC = 376	Voltage = 1.838	Tap Ohm = 4242	Law = 3675
Tap = 43	ADC = 380	Voltage = 1.857	Tap Ohm = 4343	Law = 3715
Tap = 44	ADC = 385	Voltage = 1.882	Tap Ohm = 4444	Law = 3763
Tap = 45	ADC = 388	Voltage = 1.896	Tap Ohm = 4545	Law = 3793
Tap = 46	ADC = 393	Voltage = 1.921	Tap Ohm = 4646	Law = 3842
Tap = 47	ADC = 395	Voltage = 1.931	Tap Ohm = 4747	Law = 3861
Tap = 48	ADC = 400	Voltage = 1.955	Tap Ohm = 4848	Law = 3910
Tap = 49	ADC = 402	Voltage = 1.965	Tap Ohm = 4949	Law = 3930
Tap = 50	ADC = 407	Voltage = 1.989	Tap Ohm = 5050	Law = 3978
Tap = 51	ADC = 413	Voltage = 2.019	Tap Ohm = 5151	Law = 4037
Tap = 52	ADC = 414	Voltage = 2.023	Tap Ohm = 5252	Law = 4047
Tap = 53	ADC = 419	Voltage = 2.048	Tap Ohm = 5353	Law = 4096
Tap = 54	ADC = 423	Voltage = 2.067	Tap Ohm = 5454	Law = 4135
Tap = 55	ADC = 424	Voltage = 2.072	Tap Ohm = 5555	Law = 4145
Tap = 56	ADC = 429	Voltage = 2.097	Tap Ohm = 5656	Law = 4194
Tap = 57	ADC = 433	Voltage = 2.116	Tap Ohm = 5757	Law = 4233
Tap = 58	ADC = 438	Voltage = 2.141	Tap Ohm = 5858	Law = 4282
Tap = 59	ADC = 441	Voltage = 2.155	Tap Ohm = 5959	Law = 4311

Table A.6: Digital potentiometer values on ascent 60 to 99

WiperDirection UP				
Tap = 60	ADC = 446	Voltage = 2.180	Tap Ohm = 6060	Law = 4360
Tap = 61	ADC = 449	Voltage = 2.195	Tap Ohm = 6161	Law = 4389
Tap = 62	ADC = 453	Voltage = 2.214	Tap Ohm = 6262	Law = 4428
Tap = 63	ADC = 457	Voltage = 2.234	Tap Ohm = 6363	Law = 4467
Tap = 64	ADC = 463	Voltage = 2.263	Tap Ohm = 6464	Law = 4526
Tap = 65	ADC = 466	Voltage = 2.278	Tap Ohm = 6565	Law = 4555
Tap = 66	ADC = 473	Voltage = 2.312	Tap Ohm = 6666	Law = 4624
Tap = 67	ADC = 476	Voltage = 2.326	Tap Ohm = 6767	Law = 4653
Tap = 68	ADC = 483	Voltage = 2.361	Tap Ohm = 6868	Law = 4721
Tap = 69	ADC = 487	Voltage = 2.380	Tap Ohm = 6969	Law = 4761
Tap = 70	ADC = 489	Voltage = 2.390	Tap Ohm = 7070	Law = 4780
Tap = 71	ADC = 499	Voltage = 2.439	Tap Ohm = 7171	Law = 4878
Tap = 72	ADC = 503	Voltage = 2.458	Tap Ohm = 7272	Law = 4917
Tap = 73	ADC = 510	Voltage = 2.493	Tap Ohm = 7373	Law = 4985
Tap = 74	ADC = 515	Voltage = 2.517	Tap Ohm = 7474	Law = 5034
Tap = 75	ADC = 523	Voltage = 2.556	Tap Ohm = 7575	Law = 5112
Tap = 76	ADC = 529	Voltage = 2.586	Tap Ohm = 7676	Law = 5171
Tap = 77	ADC = 537	Voltage = 2.625	Tap Ohm = 7777	Law = 5249
Tap = 78	ADC = 546	Voltage = 2.669	Tap Ohm = 7878	Law = 5337
Tap = 79	ADC = 553	Voltage = 2.703	Tap Ohm = 7979	Law = 5406
Tap = 80	ADC = 561	Voltage = 2.742	Tap Ohm = 8080	Law = 5484
Tap = 81	ADC = 570	Voltage = 2.786	Tap Ohm = 8181	Law = 5572
Tap = 82	ADC = 581	Voltage = 2.840	Tap Ohm = 8282	Law = 5679
Tap = 83	ADC = 592	Voltage = 2.893	Tap Ohm = 8383	Law = 5787
Tap = 84	ADC = 602	Voltage = 2.942	Tap Ohm = 8484	Law = 5885
Tap = 85	ADC = 613	Voltage = 2.996	Tap Ohm = 8585	Law = 5992
Tap = 86	ADC = 628	Voltage = 3.069	Tap Ohm = 8686	Law = 6139
Tap = 87	ADC = 641	Voltage = 3.133	Tap Ohm = 8787	Law = 6266
Tap = 88	ADC = 658	Voltage = 3.216	Tap Ohm = 8888	Law = 6432
Tap = 89	ADC = 673	Voltage = 3.289	Tap Ohm = 8989	Law = 6579
Tap = 90	ADC = 691	Voltage = 3.377	Tap Ohm = 9090	Law = 6755
Tap = 91	ADC = 711	Voltage = 3.475	Tap Ohm = 9191	Law = 6950
Tap = 92	ADC = 733	Voltage = 3.583	Tap Ohm = 9292	Law = 7165
Tap = 93	ADC = 756	Voltage = 3.695	Tap Ohm = 9393	Law = 7390
Tap = 94	ADC = 784	Voltage = 3.832	Tap Ohm = 9494	Law = 7664
Tap = 95	ADC = 815	Voltage = 3.983	Tap Ohm = 9595	Law = 7967
Tap = 96	ADC = 849	Voltage = 4.150	Tap Ohm = 9696	Law = 8299
Tap = 97	ADC = 889	Voltage = 4.345	Tap Ohm = 9797	Law = 8690
Tap = 98	ADC = 939	Voltage = 4.589	Tap Ohm = 9898	Law = 9179
Tap = 99	ADC = 997	Voltage = 4.873	Tap Ohm = 10000	Law = 9746

Appendix B

Prototype Construction

To start the construction, the metal box was polished and the interior shelf has removed since it was warped. Without the shelf, the rest of the scrap metal became the shell of the prototype, ensuring enough resistance to mechanical stresses.

Since there was the need to attach screws under the box, it was necessary to incorporate legs that increased the height of the box 5cm. Then, to allow an easy fixation of the components and to avoid drilling a hole for every component, wood was added to the interior. The wood, used in ceiling insulation, was, from the available options, the easiest one to manipulate without deforming under the weight from the components.

Figure B.1: Second Step of the prototype

With legs and a wood interior, the exterior paint was added. Since the exterior shell had rust, a special paint was used to clear the rust and conserve its state.

Since the work requires a Kinect and a galvanometer, which are components that need to see and project respectively, two squared holes were opened with a grinding wheel, that was the only tool available that could produce the intended result. To perform the holes, for the Kinect, the dimensions were copied from the 3D box dimensioned, as for

the galvanometer, the dimension of the holes was calculated by fixing the galvanometer on the final place and sending 5V, which is the maximum voltage, into each direction of each axis.

Figure B.2: Box with holes and paint

With the box shape in the final stages, the interiors were then developed. A three meters' power extension was added and fixed on the box using nuts outside the box shell to clamp the screw. This power extension has three electrical outlets which were meant for the Kinect, galvanometer and vibrometer. Also, a rubber washer was used to cover and protect the hole for the power extension to avoid any damage from the metal shell on the electrical cable.

Figure B.3: Box with Kinect and galvanometer

To send information to the Arduino and to receive information from the Kinect, another hole was drilled some centimeters under the power cable hole. In this hole, a rubber washer was also used to protect the cables from damage on any metal barb from the box shell.

Other eleven holes were drilled to fix the Kinect, galvanometer, laser and vibrometer. Since the box had very few light inside, a led RGB stripe was glued to the top of each shelf having the receiver of the IR signal from a remote glued, on the interior top left of the box.

The Kinect was then fixed to the box, using a made-up system using screw-bolt system and three shelf fixators. This solution keeps the Kinect, always perpendicular to the product shell, keeping its distance from the galvanometer immutable.

Figure B.4: Final look of the prototype box

The next step was the fixation of other components, and due to the lack of space on the top shelf floor, the galvanometer drivers were fixed to the interior top of the box. This created a problem, since this component is the one that creates most heat. To cool the box, a 12mm fan was then glued to the top shelf, allowing air circulation from the galvanometer hole to the Kinects hole. To power the fan and the RGB led stripe, a 12v 3A power source was used.

Figure B.5: Interior Fan

Due to the number of laser pointers used on the work, wood bases were built since its construction could be done at home at a very small cost. This wood blocks were made

from a single wood piece and cut using an electric saw. This way two laser bases were produced since the first laser pointer already had one.

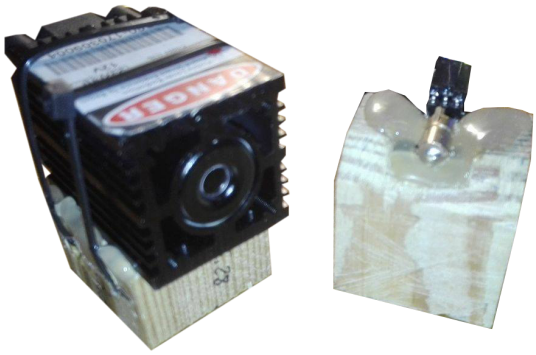


Figure B.6: Lasers

To facilitate, the detection of an object form the Kinect, a led projector was incorporated on the exterior of the box, having a on off button close to it. The led projector needs to be turned on and off manually since no relays were available at its placement. The led projector had its based cut to occupy a smaller space outside the box.

Figure B.7: Light off

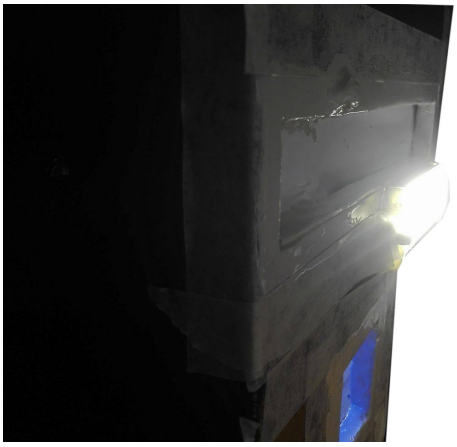


Figure B.8: Light on

To the box was then added an aluminum exterior to increase the floor width so the vibrometer could fit and to hold the back cover made from acrylic. The aluminum corners have also a track so the acrylic is kept on place. To simplify the prototype, a solution where a usb hub was installed to avoid the three cables, one from the Arduino and two from the Kinect, making it just one for the hub, this way when the user connects the

usb cable, he connects both the Kinect and the Arduino and could detach the hub cable, since it was placed outside the box, making it easier to transport, leaving just a power cable to manage. The hub solution was then not used because the Kinect only accepted official cables, so when the GUI was started, only the Arduino responded, appearing the bad connection signal on the video feed axis.

Appendix C

PWM Frequency change on the Arduino Uno

The Arduino Uno has 3 timers, timer 0, 1 and 2. The first timer, 0, controls the 5th and 6th pins, timer number 1 controls the 9th and 10th pins and timer 2 controls the 3th and 11th pins. Each timer controls the frequency of its pins so it is possible to have three different frequencies of PWM. Since time 0 controls the delay functions, its standard frequency of 976.56 Hz will not be changed. That leaves only timer 1 and 2, which standard frequencies are 490.20 Hz.[96]

The Arduino has a built-in way to change the frequency but only in a certain interval of values.[96]

PWM Frequency on Arduino Timer 1	
Divisor	Frequency Hz
1	31372.55
8	3921.16
64	490.20
256	122.55
1024	30.64

Table C.1: PWM Frequency on Arduino Timer 1

PWM Frequency on Arduino Timer 2	
Divisor	Frequency Hz
1	31372.55
8	3921.16
32	980.39
64	490.20
128	245.10
256	122.55
1024	30.64

Table C.2: PWM Frequency on Arduino Timer 2

In the previous tables, the intervals of frequency values represented which are used

on the function setPwmFrequency[64] available on the Arduino website. By observation of the values, it is possible to conclude that the desirable value of 33862.75 Hz it is bigger than the maximum possible value of this function, and the precision of the number required a function that could manipulate the Arduino frequency to any desirable value.

The alternative to the setPwmFrequency function was the PWM frequency library. This library can set the Arduino frequency to any desirable value, but it has a disadvantage, it can not change the frequency on pins 11 and 6 on the Arduino Uno. The logic gate Motorola 74HC00 mentioned before has the objective of solving this problem, since four PWM pins were necessary and only three of the non-timer0 pins were available.

From the PWM frequency library, which is called on the Arduino code with "# include<PWM.h>" [65], three functions are used.

- InitTimersSafe()
- SetPinFrequencySafe(Axispin, frequency)
- pwmWrite(pin,analogValue)

The InitTimerSafe()[65] function needs to be the first one used since it initializes all timers. Then the SetPinFrequencySafe function sets the frequency of a certain pin with the desirable value and the pwmWrite, has the same effect as the digitalWrite function of the Arduino which has as input parameters the pin and the analog value which goes from 0 to 255, which is equivalent from 0 to 5V.

Appendix D

Arduino resume

D.1 Overview of the Arduino Solution

In summary, the Arduino shields, besides controlling the galvanometer servos and controlling the laser pointer, can send IR signals and read the temperature inside the box. To accomplish these tasks, libraries had to be used since its development would represent additional work to develop something that was a secondary task on the work, being the primary goal the representation of the measurement point on an object.

To summarize the libraries used and they function, their names and function is described on the following tables.

Table D.1: Libraries used

Library name	Task
PWM.h	Change the pwm frequency on pwm pins
Adafruit_MCP4725.h	I2C control for the MCP4725 dac?s
Wire.h	To be used on the Adafruit library
IRremote.h	Control of the IR emitter
Timer.h	Timer for the temperature readings

Inside each library, only some functions were used, so they name and function is described below to allow an easier understanding of them.

Table D.2: PWM library

PWM.h [97]	
InitTimersSafe();	Start the timers
SetPinFrequencySafe(pin, frequency);	Set the pins frequency
pwmWrite(pin,value);	Set the value of the pin from 0 to 255

Table D.3: Adafruit MCP4725

Adafruit_MCP4725.h	
dac.begin(address)	Selects the slave by address
dac.setVoltage(value, false);	Set the voltage value to the selected address, then it is possible to save it on the device, "true" or not "false".

Table D.4: Irremote [102]

IRremote.h	
irsend.sendNEC(hex message, lenght);	Sends a NEC format message with a certain length with pin three on the Arduino Uno

Table D.5: Temperature timer[103]

Timer.h	
int tickEvent = t.every(interval, function)	Creates a timer that runs a function in a certain interval
t.update()	Uptades the timer

Appendix E

Devices specifications

E.1 Pro K FAI12V-3A

Table E.1: Pro K FAI12V-3A(C) specifications[22]

	Value	Unit
Input Voltage	100-240	Vac
Power	36	W
Output Voltage	12	Vdc
Current	3	A

E.2 Cooler Master DF1202512RFUN

Table E.2: Cooler Master DF1202512RFUN specifications[20]

	Value	Unit
Voltage	12	VDC
Current	0.35	A
Input	4.2	W
Speed	2000	RPM
Air pressure	2.94	mmH2O
Fan noise	19	dB-A
Fan Life Expectancy	50,000	hours

E.3 Eglo 3m RGB LED

The RGB led stripe is responsible for the interior light inside the prototype. It requires 12V to work and its control is done using an infrared command.

Table E.3: EGLO LED stripe specifications[21]

	Value	Unit
Voltage	20	V
Life expectancy	30,000	hours
Intensity	345	lm
Current	0.6	A
	IP20	

E.4 Control of the RGB lights via MATLAB

The RGB led stipe has a remote control and a IR receiver. To allow the user to control is via GUI, a remote control was built into the app with the same options as the original. The LED stipe operates with 12V, so, the power input comes from the 12V 3A power source, placed in the middle of the top shelf, the only object connected to it besides the LED's is the cooling fan.

In the GUI, each key pressed send a number to the Arduino, followed by ";", then each number corresponds to a certain code on the Arduino. The numeration is incremental, starting in one and ending in twenty-four and starts at the less brightness button on the remote control and ends in the more brightness button. The numeration is made from left to right, for each line. In order to send the message with the number, the Arduino serial connection must be on.

Figure E.1: Remote Control

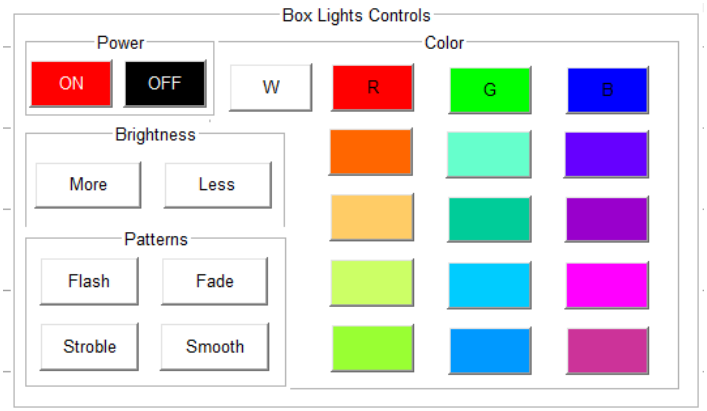


Figure E.2: App's Remote Control

E.4.1 Control of the RGB lights via Arduino

To be able to send a IR message thought the IR emitter LED available in both shields, first was necessary to be able to know which is the information the original remote is sending. First a IR receiver LED was used to read all the values of the original remote. For the reading the data pin on the led, represented in green on the next figure, was connected to pin eleven and, the library IRremote.h was used.

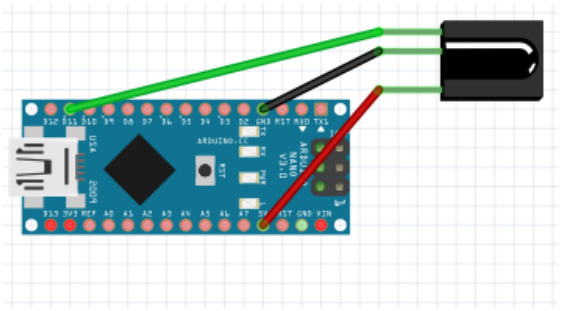


Figure E.3: IR receiver Arduino

The program used to build a IR receiver is part of the IRremote library[102]. It identifies the message protocol and size, and the hexadecimal code of the pressed button. The remote protocol is NEC and the size of the message is 32, and the decimal converted numbers of the hexadecimal code exposed in the following table.

Table E.4: Remote Control decimal code

Button	Decimal code
BTN_LED_BRILHO_ADD	16748655
BTN_LED_BRILHO_SUB	16758855
BTN_LED_OFF	16775175
BTN_LED_ON	16756815
BTN_LED_R	16750695
BTN_LED_G	16767015
BTN_LED_B	16746615
BTN_LED_W	16754775
BTN_LED_R_1	16771095
BTN_LED_G_1	16730295
BTN_LED_B_1	16738455
BTN_LED_Flash	16757325
BTN_LED_R_2	16712957
BTN_LED_G_2	16724685
BTN_LED_B_2	16720095
BTN_LED_Strobe	16711935
BTN_LED_R_3	16732335
BTN_LED_G_3	16742535
BTN_LED_B_3	16740495
BTN_LED_Fade	16734375
BTN_LED_R_4	16726215
BTN_LED_G_4	16722135
BTN_LED_B_4	16773135
BTN_LED_Smooth	16724175

With all the IR messages decrypted, the message emitter code was written. In the IRremote library, the standard emitter pin for the message is the third so it was reserved for this task when the shields were built.

To send the IR message on the Arduino, besides the importation of the IR library, a IR variable with the IR sending properties of the library was created to simplify the code, as the original was the IRsend with IR as capital letters which is then called irsend. The final step was the case switch structure to send a certain message when a certain number was received in the serial communication. Inside each case, a "irsend.sendNEC(code, LED lenght)" code was written to send the IR message with the NEC protocol, having the code as decimal or hexadecimal, since the function has an internal converter, and the led length which was a fixed value of 32.

Figure E.4: LED stripe and remote control[21]

Figure E.5: IR reciever/LED controler[21]